

ArbinCTI Manual 1.2.0 Continues01...

Feedback Handler Functions

These are call-back functions for handling the messages received from the server in response to the above-mentioned functions/APIs.

abstract void OnModifyScheduleFeedBack(ArbinCommandModifyScheduleFeed cmd);

✓ The handler for the message returned by the server in response to public bool PostModifySchedule(IArbinSocket socket, ModifyScheduleArgs args)

Input:

ArbinCommandModifyScheduleFeed cmd

```
1 public class ArbinCommandModifyScheduleFeed : ArbinCommand
2 {
3     public long TaskID { get; set; } = 0;
4
5     public List<ModifyScheduleResult> ScheduleModifyInfos { get; set; } = new List<ModifyScheduleResult>();
6 }
7
8 public class ModifyScheduleResult
9 {
10     public string ScheduleName;
11     public MODIFY_SCHEDULE_TOKEN Result;
12     public string Message;
13 }
14
15 public enum MODIFY_SCHEDULE_TOKEN
16 {
17     CTI_MODIFY_SUCCESS,
18     CTI_MODIFY_ERROR = 0x10,
19     CTI_MODIFY_SCHEDULE_NAME_EMPTY_ERROR,
20     CTI_MODIFY_SCHEDULE_NOT_EXIST_ERROR,
21     CTI_MODIFY_SCHEDULE_RUNNING_ERROR,
22     CTI_MODIFY_AUX_COUNT_EXCEED_MAPPING_ERROR,
23     CTI_MODIFY_AUX_COUNT_EXCEED_SYSCONFIG_ERROR,
24     CTI_MODIFY_AUX_SAFETY_SCOPE_ERROR,
25     CTI_MODIFY_NO_SN_CYCLER_ERROR,
26     CTI_MODIFY_NO_LOGIN_ERROR,
27     CTI_MODIFY_NO_PERMISSION_ERROR,
28 }
```

Name	DataType	Description
TaskID	long	Identifier of the command.
ScheduleModifyInfos	List<ModifyScheduleResult>	Class containing details of the schedule modifications.

abstract void OnGetMappingAuxFeedBack(ArbinCommandGetMappingAuxFeed cmd);

✓ The handler for the message returned by the server in response to public bool PostGetMappingAux(IArbinSocket socket, GetMappingAuxArgs args).

Input:

ArbinCommandGetMappingAuxFeed cmd

```
1 public class ArbinCommandGetMappingAuxFeed : ArbinCommand
2 {
3     public long TaskID { get; set; } = 0;
4
5     public List<MappingInfo> MappingInfos { get; set; } = new List<MappingInfo>();
6 }
```

Name	DataType	Description
TaskID	long	Identifier of the command.
MappingInfos	List<MappingInfo>	Class containing details of the Auxiliary channels mapped to an IV channel. Refer: Annexure L

abstract void OnLogicConnectFeedBack (ArbinCommandLogicConnectFeed cmd)

- ✓ The handler for the message returned by the server in response to bool PostLogicConnect (IArbinSocket socket, bool bSetKickOut);. Check the message to determine if user privileges have been correctly set

Input:

ArbinCommandLogicConnectFeed cmd;

```
1 public class ArbinCommandLogicConnectFeed : ArbinCommand
2 {
3     // 0: Can be kicked
4     // 1: Cannot be kicked
5     public uint dwSetKickOut;
6     // 0: Connection Accepted
7     // 1: Connection not accepted
8     public uint dwConnectResult;
9 }
```

Name	DataType	Description
dwSetKickOut	int	specifies whether the user can be kicked out of the server (0 - Yes / 1 - No)
dwConnectResult	int	specifies whether the connection request has been accepted by the server (0 - Yes / 1 - No)

i Note: A description of the ArbinCommand class can be found in Annexure I.

abstract void OnUserLoginFeedBack (ArbinCommandLoginFeed cmd)

- ✓ The callback function for the server's response to bool PostUserLogin (IArbinSocket socket, string strUser, string strPassword); Use this to determine if logging in was successful.

Input:

ArbinCommandLoginFeed cmd;

```
1 public class ArbinCommandLoginFeed : ArbinCommand
2 {
3     public enum CTI_VERSION
4     {
5         CTI_PR07_1,
6     };
7     public enum LOGIN_RESULT
8     {
9         CTI_LOGIN_SUCCESS = 1,
10        CTI_LOGIN_FAILED,
11        CTI_LOGIN_BEFORE_SUCCESS
12    };
13    public LOGIN_RESULT Result;
14    public int UserType;
15    public string SN;
16    public string Note;
17    public string NickName;
18    public string Location;
19    public string EmergencyContactNameAndPhoneNumber;
20    public string OtherComments;
21    public string Email;
22    public uint ITAC;
23    public string CALL;
24    public uint ChannelNum ;
25    public CTI_VERSION Version ;
26    public Image Img = null;
27 }
```

Valid Values for LOGIN_TOKEN Result

Name	DataType	Description
LOGIN_RESULT Result	Enum	Result contains the success message from the cyclor
CTI_LOGIN_SUCCESS		Login Success
CTI_LOGIN_FAILED		Login Failed
CTI_LOGIN_BEFORE_SUCCESS		Already Logged-in
UserType	int	Logged in user type. (admin/operator/guest)
SN	string	Serial Number of the Cyclor
Note	string	Any note added about cyclor in server
NickName	string	Nickname given to cyclor

Location	string	contains information on the cyclers location
EmergencyContactNameAndPhoneNumber	string	contains information on contact information for the person in charge of emergencies regarding the cyclers
OtherComments	string	Any comments regarding cyclers
Email	string	Email ID of the person In-charge of the Cycler
ITAC	uint	International Telephone Area Code
CALL	string	Cell phone Number
ChannelNum	uint	contains the total number of channels on the system
Img	image	contains an image set for the cyclers
CTI_VERSION Version	Enum	Version of CTI

abstract void OnAssignScheduleFeedBack (ArbinCommandAssignSchedule Feed cmd)

- ✓ The callback function for the message returned by the cyclers after sending the bool PostAssignSchedule (IArbinSocket socket, string ScheduleName, String Barcode, float Capacity, float MVUD1, float MVUD2, float MVUD3, float MVUD4, [bool AllAssign = true], [int ChannelIndex = -1]). Check the result stored in cmd to determine the status of the Assign-Schedule command.

Input:

ArbinCommandAssignScheduleFeed cmd;


```

1 public class ArbinCommandAssignScheduleFeed : ArbinCommand
2 {
3     public enum ASSIGN_TOKEN
4     {
5         CTI_ASSIGN_SUCCESS,
6         CTI_ASSIGN_FAILED,
7         CTI_ASSIGN_INDEX = 0x10,
8         CTI_ASSIGN_ERROR,
9         CTI_ASSIGN_SCHEDULE_NAME_EMPTY_ERROR,
10        CTI_ASSIGN_SCHEDULE_NOT_FIND_ERROR,
11        CTI_ASSIGN_CHANNEL_RUNNING_ERROR,
12        CTI_ASSIGN_CHANNEL_DOWNLOAD_ERROR,
13        CTI_ASSIGN_BACTH_FILE_OPENED,
14        CTI_ASSIGN_SDU_CANNOT_ASSIGN_SCHEDULE,
15        CTI_ASSIGN_SDU_SAVE_FAILED,
16        CTI_ASSIGN_FILE_UNSUPPORTED_FILE_TYPE,
17        CTI_ASSIGN_FILE_NOT_ASSIGN_SCHEDULE,
18        CTI_ASSIGN_FILE_SCHEDULE_NOT_AUX_REQUIREMENT,
19        CTI_ASSIGN_FILE_SCHEDULE_IS_RUNNING,
20        CTI_ASSIGN_SCHEDULE_MUID_NOT_SAME,
21        CTI_ASSIGN_SCHEDULE_CLEAR
22    };
23    public ASSIGN_TOKEN Result;
24 }

```

Valid Values for ASSIGN_TOKEN Result.

Enum	Description
CTI_ASSIGN_SUCCESS,	Schedule assign Successful
CTI_ASSIGN_FAILED,	Schedule assign fail
CTI_ASSIGN_INDEX = 0x10,	Channel Index error
CTI_ASSIGN_ERROR,	CTI not allowed to control
CTI_ASSIGN_SCHEDULE_NAME_EMPTY_ERROR,	Schedule Name is empty
CTI_ASSIGN_SCHEDULE__NOT_FIND_ERROR,	Schedule not found in work directory of server
CTI_ASSIGN_CHANNEL_RUNNING_ERROR,	Channel already running
CTI_ASSIGN_CHANNEL_DOWNLOAD_ERROR,	Downloading schedule
CTI_ASSIGN_BACTH_FILE_OPENED,	Batch file was opened
CTI_ASSIGN_SDU_CANNOT_ASSIGN_SCHEDULE,	assign error
CTI_ASSIGN_SDU_SAVE_FAILED,	Failed save
CTI_ASSIGN_FILE_UNSUPPORTED_FILE_TYPE,	Unsupported file type
CTI_ASSIGN_FILE_NOT_ASSIGN_SCHEDULE,	Not assign schedule
CTI_ASSIGN_FILE_SCHEDULE_NOT_AUX_REQUIREMENT,	Schedule not aux requirement
CTI_ASSIGN_FILE_SCHEDULE_IS_RUNNING,	Schedule is running
CTI_ASSIGN_SCHEDULE_MUID_NOT_SAME,	MUID of schedule file not consistent
CTI_ASSIGN_SCHEDULE_CLEAR,	Clear schedule file

 If a schedule file is copy/pasted from another machine its MUID will be inconsistent. It either needs to be created locally or uploaded from CTI.

abstract void OnSetMVFeedBack (ArbinCommandSetMetaVariableFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostSetMetaVariable (IArbinSocket socket, int nChannel, int mvType, int mvMetaCode, int mvValueType, object mvValue);

Input:

ArbinCommandSetMeatVariableFeed cmd;

```

1 public class ArbinCommandSetMetaVariableFeed: ArbinCommand
2 {
3     public enum SET_MV_RESULT
4     {
5         CTI_SET_MV_SUCCESS,
6         CTI_SET_MV_FAILED=16,
7         CTI_SET_MV_METACODE_NOTEXIST,
8         CTI_SET_MV_CHANNEL_NOT_STARTED,
9         CTI_SET_MV_METACODE_NOTEXIST_Pro7
10    };
11    public SET_MV_RESULT Result;
```

```
12     }
```

Valid Values for SET_MV_RESULT Result.

Enum	Description
CTI_SET_MV_SUCCESS	Metavariable successfully set
CTI_SET_MV_FAILED	Metavariable not set
CTI_SET_MV_METACODE_NOTEXIST	Metacode specified does not exists
CTI_SET_MV_CHANNEL_NOT_STARTED	Channel not running
CTI_SET_MV_METACODE_NOTEXIST_Pro7	Not exist pro7

abstract void OnUpdateMetaVariableAdvancedFeedBack (ArbinCommand UpdateMetaVariableAdvancedFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostUpdateMetaVariableAdvanced (IArbinSocket socket, List<MetaVariableInfo> metaVariableList, out int error);

Input:

ArbinCommandUpdateMetaVariableAdvancedFeed cmd;

```
1 public class ArbinCommandUpdateMetaVariableAdvancedFeed: ArbinCommand
2 {
3     public enum SET_MV_RESULT
4     {
5         CTI_SET_MV_SUCCESS,
6         CTI_SET_MV_FAILED=16,
7         CTI_SET_MV_METACODE_NOTEXIST,
8         CTI_SET_MV_CHANNEL_NOT_STARTED,
9         CTI_SET_MV_METACODE_UPDATE_TOO_FREQUENTLY_200MS
10    };
11    public SET_MV_RESULT Result;
12 }
```

Valid Values for SET_MV_RESULT Result.

Enum	Description
CTI_SET_MV_SUCCESS	Metavariable successfully set
CTI_SET_MV_FAILED	Metavariable not set
CTI_SET_MV_METACODE_NOTEXIST	Metacode specified does not exists
CTI_SET_MV_CHANNEL_NOT_STARTED	Channel not running
CTI_SET_MV_METACODE_UPDATE_TOO_FREQUENTLY_200MS	Too frequently 200ms

abstract void OnTimeSensitiveSetMVFeedBack(ArbinCommandTimeSensitiveSetMVFeed cmd);

- ✓ The callback function for the server's response to bool PostTimeSensitiveSetMV(IArbinSocket socket, TimeSensitiveSetMVArgs args);

Input:

ArbinCommandTimeSensitiveSetMVFeed cmd;

```
1 public class ArbinCommandTimeSensitiveSetMVFeed : ArbinCommand
2 {
3     /// <summary>
4     /// Control status
5     /// </summary>
6     public enum EControlStatus
7     {
8         /// <summary>
9         /// Channel is not being used.
10        /// </summary>
11        Idle,
12        /// <summary>
13        /// The microcontroller has delayed data acquisition while it is processing operations
associated with step changes.
14        /// </summary>
15        Transition,
16        /// <summary>
17        /// Present measured channel current is positive.
18        /// </summary>
19        Charge,
20        /// <summary>
21        /// Present measured channel current is negative.
22        /// </summary>
23        Discharge,
24        /// <summary>
25        /// The charge/discharge circuits are disconnected from the test sample, but the voltage
measurement circuit is still connected.
26        /// </summary>
27        Rest,
28        /// <summary>
29        /// MITS Pro is waiting for some condition on one or more channels to be realized before
proceeding with the schedule sequence. (used with SAFOR systems).
30        /// </summary>
31        Wait,
32        /// <summary>
33        /// disables the current and voltage control of the main IV channel and records the current
that flows through the External Charge adaptor.
34        /// </summary>
35        External_Charge,
36        /// <summary>
37        /// The channel is being calibrated and it is impossible to receive this status.
38        /// </summary>
39        Calibration,
40        /// <summary>
41        /// Value of any parameters exceeds the safety limit set in schedule.
42        /// </summary>
43        Unsafe,
44        /// <summary>
45        /// Channel is generating current or voltage pulses.
46        /// </summary>
47        Pulse,
48        /// <summary>
49        /// Channel is executing internal resistance diagnostic.
50        /// </summary>
51        Internal_Resistance,
```

```

52      ///Channel is executing AC impedance diagnostic.
54      </summary>
55      AC_Impedance,
56      ///<para><b>Don't care.</b></para>
58      </summary>
59      ACI_Cell,
60      ///CTI(Console TCP Interface) maintains a waiting condition on a channel while attempting to
        attain Test Setting Values.
62      </summary>
63      Test_Settings,
64      ///system error.
66      </summary>
67      Error,
68      ///The test has proceeded to completion and terminated according to scheduled limit
        conditions.
70      </summary>
71      Finished,
72      ///<para><b>Don't care.</b></para>
74      </summary>
75      Volt_Meter,
76      ///<para><b>Don't care.</b></para>
78      </summary>
79      Waiting_for_ACS,
80      ///The charge/discharge circuits are disconnected from the test sample, but the voltage
        measurement circuit is still connected, no data log and press continue button test will go to next step.
82      </summary>
83      Pause,
84      ///<para><b>Don't care.</b></para>
86      </summary>
87      EMPTY,
88      ///<para><b>Don't care.</b></para>
90      </summary>
91      Idle_from_MCU,
92      ///<para><b>Don't care.</b></para>
94      </summary>
95      Start,
96      ///<para><b>Don't care.</b></para>
98      </summary>
99      Running,
100     ///<para><b>Don't care.</b></para>
102     </summary>
103     Step_Transfer,
104     ///<para><b>Don't care.</b></para>
106     </summary>

```



```

107         Resume,
108         /// <summary>
109         /// <para><b>Don't care.</b></para>
110         /// </summary>
111         Go_Pause,
112         /// <summary>
113         /// <para><b>Don't care.</b></para>
114         /// </summary>
115         Go_Stop,
116         /// <summary>
117         /// <para><b>Don't care.</b></para>
118         /// </summary>
119         Go_Next_Step,
120         /// <summary>
121         /// <para><b>Don't care.</b></para>
122         /// </summary>
123         Online_Update,
124         /// <summary>
125         /// <para><b>Don't care.</b></para>
126         /// </summary>
127         Daq_Memory_Unsafe,
128         /// <summary>
129         /// <para><b>Don't care.</b></para>
130         /// </summary>
131         ACR,
132         /// <summary>
133         /// <para><b>Don't care.</b></para>
134         /// </summary>
135         CS_SUSPENT
136     };
137
138     /// <summary>
139     /// Control status
140     /// </summary>
141     public enum EResult
142     {
143         /// <summary>
144         /// Success
145         /// </summary>
146         SUCCESS,
147         /// <summary>
148         /// Success but no Running for the channel
149         /// </summary>
150         SUCCESS_NOTRUNNING,
151         /// <summary>
152         /// Execute error
153         /// </summary>
154         ERROR = 0x10,
155         /// <summary>
156         /// Data type not support
157         /// </summary>
158         DATATYPE_NOTSUPPORT,
159         /// <summary>
160         /// Metacode not exist
161         /// </summary>
162         METACODE_NOTEXIST,
163         /// <summary>
164         /// Channel index error

```

```

165         /// <summary>
166         CHANNEL_INDEX_ERROR,
167         /// <summary>
168         /// Auxiliary index error
169         /// </summary>
170         AUX_INDEX_ERROR,
171         /// <summary>
172         /// Auxiliary not assign
173         /// </summary>
174         AUX_NOTASSIGN,
175         /// <summary>
176         /// CANBMS index error
177         /// </summary>
178         CANBMS_INDEX_ERROR,
179         /// <summary>
180         /// CANBMS not exist
181         /// </summary>
182         CANBMS_NOTEXIST,
183         /// <summary>
184         /// CANBMS disabled
185         /// </summary>
186         CANBMS_DISABLED,
187         /// <summary>
188         /// Not connect Daq
189         /// </summary>
190         NOT_CONNECT_DAQ,
191         /// <summary>
192         /// Timeout
193         /// </summary>
194         TIMEOUT,
195         /// <summary>
196         /// MCU ACK failed
197         /// </summary>
198         MCU_ACK_FAILED,
199         /// <summary>
200         /// Not allow control
201         /// </summary>
202         NOT_ALLOW_CONTROL,
203         /// <summary>
204         /// MCU Socket Disconnected
205         /// </summary>
206         MCU_SOCKET_DISCONNECTED
207     };
208
209     /// <summary>
210     /// Result
211     /// </summary>
212     public class TimeSensitiveSetMVResult
213     {
214         /// <summary>
215         /// IV Channel Global Index, starts from zero.
216         /// </summary>
217         public int GlobalIndex;
218
219         public int StepIndex { get; set; } = -1;
220         public int SubStepIndex { get; set; } = -1;
221
222         /// <summary>

```

```

223     /// Machine Status
224     /// </summary>
225     public EControlStatus MachineStatus;
226
227     /// <summary>
228     /// Result
229     /// </summary>
230     public EResult Result;
231
232     /// <summary>
233     /// Current value after setting MV
234     /// </summary>
235     public float Current;
236
237     /// <summary>
238     /// Voltage value after setting MV
239     /// </summary>
240     public float Voltage;
241
242     public List<TimeSensitiveSetMV> MVs { get; internal set; } = new List<TimeSensitiveSetMV>();
243 }
244
245 public List<TimeSensitiveSetMVResult> Results = new List<TimeSensitiveSetMVResult>();
246 }

```

Valid Values for EResult.

Enum	Enum Value(int)	Description
SUCCESS	0	Successful
SUCCESS_NOTRUNNING	1	Success but no Running for the channel
ERROR	16	Execute error
DATATYPE_NOTSUPPORT	17	Data type not support
METACODE_NOTEXIST	18	Metacode not exist
CHANNEL_INDEX_ERROR	19	Channel index error
AUX_INDEX_ERROR	20	Auxiliary index error
AUX_NOTASSIGN	21	Auxiliary not assign
CANBMS_INDEX_ERROR	22	CANBMS index error
CANBMS_NOTEXIST	23	CANBMS not exist
CANBMS_DISABLED	24	CANBMS disabled
NOT_CONNECT_DAQ	25	Not connect Daq
TIMEOUT	26	Timeout
MCU_ACK_FAILED	27	MCU ACK failed
NOT_ALLOW_CONTROL	28	Not allow control, "Allow control" is unchecked in the user configuration

abstract void OnGetMetaVariablesFeedBack (ArbinCommand ArbinCommandGetMetaVariablesFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostGetMetaVariables(IArbinSocket socket, List<ArbinCommandGetMetaVariablesFeed.MetaVariableInfo> metaVariableList, out int error);

Input:

ArbinCommandGetMetaVariablesFeed cmd;

```
1 public class ArbinCommandGetMetaVariablesFeed: ArbinCommand
2 {
3     public enum GET_MV_RESULT
4     {
5         CTI_GET_MV_SUCCESS = 0x0,
6         CTI_GET_MV_ERROR = 0x10,
7         CTI_GET_MV_DATATYPE_NOTSUPPORT,
8         CTI_GET_MV_METACODE_NOTEXIST,
9         CTI_GET_MV_CHANNEL_INDEX_ERROR,
10        CTI_GET_MV_AUX_INDEX_ERROR,
11        CTI_GET_MV_AUX_NOTASSIGN,
12        CTI_GET_MV_CANBMS_INDEX_ERROR,
13        CTI_GET_MV_CANBMS_NOTEXIST,
14        CTI_GET_MV_CANBMS_DISABLED,
15        CTI_GET_MV_NOT_CONNECT_DAQ,
16        CTI_GET_MV_TIMEOUT,
17        CTI_GET_MV_MCU_ACK_FAILED,
18        CTI_GET_MV_METACODE_NOTSUPPORT,
19        CTI_GET_MV_SMB_NOTEXIST,
20        CTI_GET_MV_SMB_INDEX_ERROR,
21        CTI_GET_MV_SMB_NOTSUPPORT_STRING,
22        CTI_GET_MV_SMB_DISABLED,
23        CTI_GET_MV_AUX_TYPE_ERROR,
24        CTI_GET_MV_OBJ_NULL_ERROR,
25        CTI_GET_MV_DCOM_ERROR,
26        CTI_GET_MV_WRITE_NOT_SUPPORTED,
27        CTI_GET_MV_EQ_INDEX_ERROR,
28        CTI_GET_MV_CELL_INDEX_ERROR
29    }
30    public class MetaVariableInfo
31    {
32        public ushort m_Channel; //Channel/EQ/CELL Global Index (starting at 0)
33        public GET_MV_RESULT m_MV_Error;
34        public TE_DATA_TYPE m_MV_DataType;
35        public ushort m_MV_MetaCode;
36        public float m_Value;
37    }
38    public List<MetaVariableInfo> MetaVariableInfos = new List<MetaVariableInfo>();
39 }
```

Valid Values for GET_MV_RESULT.

Enum	Description
CTI_GET_MV_SUCCESS	Get meta variables successful
CTI_GET_MV_ERROR	Get meta variables error

CTI_GET_MV_DATATYPE_NOTSUPPORT	Unsupported type
CTI_GET_MV_METACODE_NOTEXIST	Meta code does not exist
CTI_GET_MV_CHANNEL_INDEX_ERROR	channel index error
CTI_GET_MV_AUX_INDEX_ERROR	Auxiliary index error
CTI_GET_MV_AUX_NOTASSIGN	Auxiliary not assigned to channel
CTI_GET_MV_CANBMS_INDEX_ERROR	Canbms index error
CTI_GET_MV_CANBMS_NOTEXIST	CAMBMS does not exist
CTI_GET_MV_CANBMS_DISABLED	Not enabled in the CAMBMS file
CTI_GET_MV_NOT_CONNECT_DAO	Can't connect to DAQ
CTI_GET_MV_TIMEOUT	Time out
CTI_GET_MV_MCU_ACK_FAILED	MCU ack failed
CTI_GET_MV_METACODE_NOTSUPPORT	Metacode Notsupport
CTI_GET_MV_SMB_NOTEXIST	SMB does not exist
CTI_GET_MV_SMB_INDEX_ERROR	SMB index error
CTI_GET_MV_SMB_NOTSUPPORT_STRING	Not support string
CTI_GET_MV_SMB_DISABLED	Not enabled in the SMB file
CTI_GET_MV_SMB_NOTEXIST	SMB does not exist
CTI_GET_MV_SMB_INDEX_ERROR	SMB index error
CTI_GET_MV_SMB_NOTSUPPORT_STRING	Not support string
CTI_GET_MV_SMB_DISABLED	Not enabled in the SMB file
CTI_GET_MV_AUX_TYPE_ERROR	Auxiliary type error
CTI_GET_MV_OBJ_NULL_ERROR	Object is null
CTI_GET_MV_DCOM_ERROR	DCOM error
CTI_GET_MV_WRITE_NOT_SUPPORTED	Write not supported
CTI_GET_MV_EQ_INDEX_ERROR	EQ index error
CTI_GET_MV_CELL_INDEX_ERROR	Cell index error

abstract void OnSetIntervalTimeLogDataFeedBack (ArbinCommandSetIntervalTimeLogDataFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostSetIntervalTimeLogData(IArbinSocket socket, float fIntervalTime);

Input:

ArbinCommandSetIntervalTimeLogDataFeed cmd;

```

1 public class ArbinCommandSetIntervalTimeLogDataFeed: ArbinCommand
2 {

```

```

3      public enum SET_INTERVAL_TIME_LOG_DATA_RESULT
4      {
5          SET_INTERVALTIME_LOGDATA_DAO_DISCONNECTED = -2,
6          SET_INTERVALTIME_LOGDATA_DCOM_FAILED = -1,
7          SET_INTERVALTIME_LOGDATA_SUCCESS = 0,
8          SET_INTERVALTIME_LOGDATA_PARTIAL_SUCCESS = 1,
9          SET_INTERVALTIME_LOGDATA_FAILED = 2,
10         SET_INTERVALTIME_LOGDATA_EXCEED_LIMIT = 3
11     }
12     public SET_INTERVAL_TIME_LOG_DATA_RESULT Result =
13     SET_INTERVAL_TIME_LOG_DATA_RESULT.SET_INTERVALTIME_LOGDATA_SUCCESS;
14     public string Message;
15 }

```

Valid Values for SET_INTERVAL_TIME_LOG_DATA_RESULT.

Enum	Description
SET_INTERVALTIME_LOGDATA_DAO_DISCONNECTED	Daq Disconnected
SET_INTERVALTIME_LOGDATA_DCOM_FAILED	DCOM Failed
SET_INTERVALTIME_LOGDATA_SUCCESS	Success
SET_INTERVALTIME_LOGDATA_PARTIAL_SUCCESS	Partial success
SET_INTERVALTIME_LOGDATA_FAILED	Failed
SET_INTERVALTIME_LOGDATA_EXCEED_LIMIT	Failed

abstract void OnApplyForUDPCommunicationFeedBack (ArbinCommandApplyForUDPCommunicationFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostApplyForUDPCommunication(IArbinSocket socket, ushort uUDPClientPort, string strUDPClientIP, List<CMetavariableDataCodeApply> dataCodeApplies);

Input:

ArbinCommandApplyForUDPCommunicationFeed cmd;

```

1  public class ArbinCommandApplyForUDPCommunicationFeed : ArbinCommand
2  {
3      public enum APPLY_UDP_COMMUNICATIO_RESULT
4      {
5          SET_INTERVALTIME_LOGDATA_DAO_DISCONNECTED = -2,
6          CTI_APPLY_UDP_COMMUNICATIO_SUCCESS = 0x0,
7          CTI_APPLY_UDP_COMMUNICATIO_ERROR = 0x10,
8          CTI_APPLY_UDP_COMMUNICATIO_DATATYPE_NOTSUPPORT,
9          CTI_APPLY_UDP_COMMUNICATIO_METACODE_NOTEXIST,
10         CTI_APPLY_UDP_COMMUNICATIO_CHANNEL_INDEX_ERROR,
11         CTI_APPLY_UDP_COMMUNICATIO_AUX_INDEX_ERROR,
12         CTI_APPLY_UDP_COMMUNICATIO_AUX_NOTASSIGN,
13         CTI_APPLY_UDP_COMMUNICATIO_CANBMS_INDEX_ERROR,
14         CTI_APPLY_UDP_COMMUNICATIO_CANBMS_NOTEXIST,
15         CTI_APPLY_UDP_COMMUNICATIO_CANBMS_DISABLED,
16         CTI_APPLY_UDP_COMMUNICATIO_NOT_CONNECT_DAO,
17         CTI_APPLY_UDP_COMMUNICATIO_TIMEOUT,
18         CTI_APPLY_UDP_COMMUNICATIO_MCU_ACK_FAILED,

```

```

19         CTI_APPLY_UDP_COMMUNICATIO_METACODEE_NOTSUPPORT,
20         CTI_APPLY_UDP_COMMUNICATIO_SMB_NOTEXIST,
21         CTI_APPLY_UDP_COMMUNICATIO_SMB_INDEX_ERROR,
22         CTI_APPLY_UDP_COMMUNICATIO_SMB_NOTSUPPORT_STRING,
23         CTI_APPLY_UDP_COMMUNICATIO_SMB_DISABLED,
24         CTI_APPLY_UDP_COMMUNICATIO_AUX_TYPE_ERROR,
25         CTI_APPLY_UDP_COMMUNICATIO_OBJ_NULL_ERROR,
26         CTI_APPLY_UDP_COMMUNICATIO_DCOM_ERROR,
27         CTI_APPLY_UDP_COMMUNICATIO_WRITE_NOT_SUPPORTED,
28         CTI_APPLY_UDP_COMMUNICATIO_EQ_INDEX_ERROR,
29         CTI_APPLY_UDP_COMMUNICATIO_CELL_INDEX_ERROR,
30     }
31     public class MetaVariableInfo
32     {
33         public APPLY_UDP_COMMUNICATIO_RESULT Error;
34         public TE_DATA_TYPE MVType;
35         public ushort MetaCode;
36         public ushort GlobalIndex;
37         public string MCUIP;
38         public ushort MCUPort;
39         public EReadWriteMode ReadWriteMode;
40     }
41     public List<MetaVariableInfo> MetaVariableInfos = new List<MetaVariableInfo>();
42 }

```

Valid Values for APPLY_UDP_COMMUNICATIO_RESULT.

Enum	Description
CTI_APPLY_UDP_COMMUNICATIO_SUCCESS	Successful
CTI_APPLY_UDP_COMMUNICATIO_ERROR	Error
CTI_APPLY_UDP_COMMUNICATIO_DATATYPE_NOTSUPPORT	Data type not support
CTI_APPLY_UDP_COMMUNICATIO_METACODE_NOTEXIST	Metacode not exist
CTI_APPLY_UDP_COMMUNICATIO_CHANNEL_INDEX_ERROR	Channel index error
CTI_APPLY_UDP_COMMUNICATIO_AUX_INDEX_ERROR	Auxiliary index error
CTI_APPLY_UDP_COMMUNICATIO_AUX_NOTASSIGN	Auxiliary not assign
CTI_APPLY_UDP_COMMUNICATIO_CANBMS_INDEX_ERROR	CANBMS index error
CTI_APPLY_UDP_COMMUNICATIO_CANBMS_NOTEXIST	CANBMS not exist
CTI_APPLY_UDP_COMMUNICATIO_CANBMS_DISABLED	CANBMS disabled
CTI_APPLY_UDP_COMMUNICATIO_NOT_CONNECT_DAQ	Not connect Daq
CTI_APPLY_UDP_COMMUNICATIO_TIMEOUT	Timeout
CTI_APPLY_UDP_COMMUNICATIO_MCU_ACK_FAILED	MCU ACK failed
CTI_APPLY_UDP_COMMUNICATIO_METACODEE_NOTSUPPORT	Meta code not support
CTI_APPLY_UDP_COMMUNICATIO_SMB_NOTEXIST	SMB not exist
CTI_APPLY_UDP_COMMUNICATIO_SMB_INDEX_ERROR	SMB index error

CTI_APPLY_UDP_COMMUNICATIO_SMB_NOTSUPPORT_STRING	SMB Not Support String
CTI_APPLY_UDP_COMMUNICATIO_SMB_DISABLED	SMB disabled
CTI_APPLY_UDP_COMMUNICATIO_AUX_TYPE_ERROR	Auxiliary type error
CTI_APPLY_UDP_COMMUNICATIO_OBJ_NULL_ERROR	Object is null
CTI_APPLY_UDP_COMMUNICATIO_DCOM_ERROR	DCOM error
CTI_APPLY_UDP_COMMUNICATIO_WRITE_NOT_SUPPORTED	Write error
CTI_APPLY_UDP_COMMUNICATIO_EQ_INDEX_ERROR	EQ index error
CTI_APPLY_UDP_COMMUNICATIO_CELL_INDEX_ERROR	Cell index error

abstract void UpdateParametersBack(ArbinCommandUpdateParameterFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostUpdateParameters(IArbinSocket socket, List<ushort> chanList, bool isForAllChans, SortedDictionary<ushort, double> parameters)

Input:

ArbinCommandUpdateParameterFeed cmd;

```

1 public class ArbinCommandUpdateParameterFeed : ArbinCommand
2     {
3         public enum UPDATE_TOKEN
4         {
5             CTI_UPDATE_SUCCESS,
6             CTI_UPDATE_INDEX = 0x10,
7             CTI_UPDATE_ERROR,
8             CTI_UPDATE_TESTOBJECT_NAME_EMPTY_ERROR,
9             CTI_UPDATE_TESTOBJECT_NOT_FIND_ERROR,
10            CTI_UPDATE_CHANNEL_RUNNING_ERROR,
11            CTI_UPDATE_CHANNEL_DOWNLOAD_ERROR,
12            CTI_UPDATE_BACTH_FILE_OPENED,
13            CTI_UPDATE_TO_CANNOT_ASSIGN_TESTOBJECT,
14            CTI_UPDATE_TESTOBJECT_SAVE_FAILED,
15        };
16        public enum EParameterDataType : ushort
17        {
18            NormCapacity = 0,
19            IMax,
20            VMax,
21            VMin,
22            Mass,
23            SCapacity,
24            NIR,
25            NVoltage,
26            NCapacitance,
27            IsAutoCalculate
28        }
29        public string Reason;
30        public UPDATE_TOKEN Result;
31    }

```

Valid Values for UPDATE_RESULT.

Enum	Description
CTI_UPDATE_SUCCESS	successful
CTI_UPDATE_INDEX	Channel Index error
CTI_UPDATE_ERROR	Execute error
CTI_UPDATE_TESTOBJECT_NAME_EMPTY_ERROR	Schedule Name is empty
CTI_UPDATE_TESTOBJECT_NOT_FIND_ERROR	Not find Schedule
CTI_UPDATE_CHANNEL_RUNNING_ERROR	Channel Running
CTI_UPDATE_CHANNEL_DOWNLOAD_ERROR	Downloading schedule
CTI_UPDATE_BACTH_FILE_OPENED	Batch File opened
CTI_UPDATE_TO_CANNOT_ASSIGN_TESTOBJECT	Assign error
CTI_UPDATE_TESTOBJECT_SAVE_FAILED	Failed save
CTI_GET_MV_NOT_CONNECT_DAQ	Can't connect to DAQ

Valid Values for EParameterDataType.

Enum	Description
NormCapacity	Nominal Capacity
IMax	Current maximum
VMax	Voltage maximum
VMin	Voltage minimum
Mass	Mass
SCapacity	Special Capacity
NIR	Nominal IR
NVoltage	Nominal voltage
NCapacitance	Nominal Capacitance
IsAutoCalculate	Is auto calculate

abstract void OnConvertToAnonymousOrNamedTOBack(ArbinCommandConvertToAnonymousOrNamedTOFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostConvertToAnonymousOrNamedTO(IArbinSocket socket, bool toAnonymous, bool isForAll, List<ushort> channelList);

Input:

ArbinCommandConvertToAnonymousOrNamedTOFeed cmd;

```

1 public class (ArbinCommandConvertToAnonymousOrNamedTOFeed : ArbinCommand
2 {

```

```

3      public enum CONVERTTESTOBJECT_RESULT
4      {
5          CTI_CONVERT_SUCCESS,
6          CTI_CONVERT_FAILED,
7      }
8
9      public string Reason;
10     public CONVERTTESTOBJECT_RESULT Result;
11     public SortedDictionary<CONVERTTESTOBJECT_RESULT, List<int>> ResultChanListPairs;
12 }

```

Valid Values for CONVERTTESTOBJECT_RESULT.

Enum	Description
CTI_CONVERT_SUCCESS	successful
CTI_CONVERT_FAILED	failed

abstract void OnAssignBarcodeInfoFeedBack(ArbinCommandAssignBarcodeFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostAssignBarcodeInfo(IArbinSocket socket, EChannelType channelType, List<ChannelBarcodeInfo> barcodeInfos),

Input:

ArbinCommandAssignBarcodeInfoFeed cmd;

```

1  public class ArbinCommandAssignBarcodeInfoFeed : ArbinCommand
2  {
3      public enum ASSIGN_BARCODE_RESULT
4      {
5          CTI_ASSIGN_BARCODE_SUCCESS,
6          CTI_ASSIGN_BARCODE_ERROR,
7          CTI_ASSIGN_BARCODE_CHANNEL_RUNNING,
8          CTI_ASSIGN_BARCODE_CHANNEL_INDEX,
9          CTI_ASSIGN_BARCODE_CHANNEL_TYPE_NOT_SUPPORT,
10         CTI_ASSIGN_BARCODE_CELL_INDEX,
11         CTI_ASSIGN_BARCODE_EQ_INDEX,
12         CTI_ASSIGN_BARCODE_TRAY_INDEX,
13         CTI_ASSIGN_BARCODE_SPTTMAPPING,
14     }
15
16     public enum EChannelType
17     {
18         IV = 1,
19         EQ = 2,
20         CELL = 3,
21         TRAY = 4
22     };
23
24     public class ChannelBarcodeInfo
25     {
26         /// <summary>
27         /// IV Global Index/EQ Global Index/Cell Global Index/Tray Global Index
28         /// </summary>
29         public ushort GlobalIndex;
30         /// <summary>

```

```

31      /// Barcode
32      /// </summary>
33      public string Barcode;
34      /// <summary>
35      /// Info
36      /// </summary>
37      public string Info;
38      /// <summary>
39      /// Error Code
40      /// </summary>
41      public ASSIGN_BARCODE_RESULT Error;
42  }
43
44      /// <summary>
45      /// Barcode Info List
46      /// </summary>
47      public List<ChannelBarcodeInfo> BarcodeInfos = new List<ChannelBarcodeInfo>();
48
49      /// <summary>
50      /// Channel Type
51      /// </summary>
52      public EChannelType ChannelType = EChannelType.IV;
53  }

```

Valid Values for ASSIGN_BARCODE_RESULT.

Enum	Description
CTI_ASSIGN_BARCODE_SUCCESS	successful
CTI_ASSIGN_BARCODE_ERROR	failed
CTI_ASSIGN_BARCODE_CHANNEL_RUNNING	Channel is running
CTI_ASSIGN_BARCODE_CHANNEL_INDEX	Channel index error
CTI_ASSIGN_BARCODE_CHANNEL_TYPE_NOT_SUPPORT	Channel type not support error
CTI_ASSIGN_BARCODE_CELL_INDEX	Cell index error
CTI_ASSIGN_BARCODE_EQ_INDEX	EQ index error
CTI_ASSIGN_BARCODE_TRAY_INDEX	Tray index error
CTI_ASSIGN_BARCODE_SPTTMAPPING	SPTT mapping;IV not assign EQ

abstract void OnGetBarcodeInfoFeedBack(ArbinCommandGetBarcodeInfoFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostGetBarcodeInfo(IArbinSocket socket, EChannelType channelType, List<GetChannelBarcodeInfo> barcodeInfos),

Input:

ArbinCommandGetBarcodeInfoFeed cmd;

```

1  public class ArbinCommandGetBarcodeInfoFeed : ArbinCommand
2  {
3      public enum GET_BARCODE_RESULT
4      {
5          CTI_GET_BARCODE_SUCCESS,

```

```

6      CTI_GET_BARCODE_ERROR = 0x10,
7      CTI_GET_BARCODE_CHANNEL_INDEX,
8      CTI_GET_BARCODE_CHANNEL_TYPE_NOT_SUPPORT,
9      CTI_GET_BARCODE_CELL_INDEX,
10     CTI_GET_BARCODE_EQ_INDEX,
11     CTI_GET_BARCODE_TRAY_INDEX,
12     CTI_GET_BARCODE_SPTTMAPPING,
13 }
14
15 public enum EChannelType
16 {
17     IV = 1,
18     EQ = 2,
19     CELL = 3,
20     TRAY = 4
21 };
22
23 /// <summary>
24 /// Get Barcode Infos
25 /// </summary>
26 public class GetChannelBarcodeInfo
27 {
28     /// <summary>
29     /// IV Global Index/EQ Global Index/Cell Global Index/Tray Global Index
30     /// </summary>
31     public ushort GlobalIndex;
32 }
33
34 /// <summary>
35 /// For get barcode
36 /// </summary>
37 public class ChannelBarcodeInfo
38 {
39     /// <summary>
40     /// IV Global Index/EQ Global Index/Cell Global Index/Tray Global Index
41     /// </summary>
42     public ushort GlobalIndex;
43     /// <summary>
44     /// Barcode
45     /// </summary>
46     public string Barcode;
47     /// <summary>
48     /// Info
49     /// </summary>
50     public string Info;
51     /// <summary>
52     /// Error Code
53     /// </summary>
54     public GET_BARCODE_RESULT Error;
55 }
56
57 /// <summary>
58 /// Barcode Info List
59 /// </summary>
60 public List<ChannelBarcodeInfo> BarcodeInfos = new List<ChannelBarcodeInfo>();
61
62 /// <summary>
63 /// Channel Type

```

```

64     /// </summary>
65     public EChannelType ChannelType = EChannelType.IV;
66 }

```

Valid Values for GET_BARCODE_RESULT.

Enum	Description
CTI_GET_BARCODE_SUCCESS	successful
CTI_GET_BARCODE_ERROR	failed
CTI_GET_BARCOD_CHANNEL_INDEX	Channel index error
CTI_GET_BARCODE_CHANNEL_TYPE_NOT_SUPPORT	Channel type not support error
CTI_GET_BARCODE_CELL_INDEX	Cell index error
CTI_GET_BARCODE_EQ_INDEX	EQ index error
CTI_GET_BARCODE_TRAY_INDEX	Tray index error
CTI_GET_BARCODE_SPTTMAPPING	SPTT mapping;IV not assign EQ

abstract void OnGetMachineTypeFeedBack(ArbinCommandGetMachineTypeFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool bool PostGetMachineType(IArbinSocket socket).

Input:

ArbinCommandGetMachineTypeFeed cmd;

```

1 public class ArbinCommandGetMachineTypeFeed : ArbinCommand
2 {
3     public enum GET_MACHINE_RESULT
4     {
5         CTI_GET_MACHINE_SUCCESS,
6         CTI_GET_MACHINE_ERROR = 0x10,
7     }
8
9     public enum EMachineType
10    {
11        IV = 1,
12        SPTT = 2,
13        //4
14        //8
15
16        ALL = IV | SPTT,
17    };
18
19    public GET_MACHINE_RESULT m_Error;
20
21    public EMachineType MachineType;
22 }

```

Valid Values for GET_MACHINE_RESULT.

Enum	Description
------	-------------

CTI_GET_MACHINE_SUCCESS	successful
CTI_GET_MACHINE_ERROR	failed

abstract void OnGetTrayStatusFeedBack(ArbinCommandGetTrayStatusFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostGetTrayStatus(IArbinSocket socket, List<int> trayIndexList, bool bSubscribe, out int error),

Input:

ArbinCommandGetTrayStatusFeed cmd;

```

1  public class ArbinCommandGetTrayStatusFeed : ArbinCommand
2      {
3      public enum GET_TRAY_STATUS_RESULT
4      {
5          CTI_GET_TRAY_STATUS_SUCCESS,
6          CTI_GET_TRAY_STATUS_ERROR,
7          CTI_GET_TRAY_INDEX_ERROR,
8          CTI_GET_TRAY_STATUS_ERROR_NETWORK,
9          CTI_GET_TRAY_STATUS_ERROR_UNIT,
10         CTI_GET_TRAY_STATUS_ERROR_SPTTMAPPING_INDEX,
11     }
12
13     public enum ETrayStatus
14     {
15         Engaged = 0,
16         Disengaged,
17         Malfunction,
18     }
19
20     public class TrayStatusInfo
21     {
22         /// <summary>
23         /// Tray Global Index(starting at 0)
24         /// </summary>
25         public ushort TrayIndex;
26         /// <summary>
27         /// Error
28         /// </summary>
29         public GET_TRAY_STATUS_RESULT Error;
30         /// <summary>
31         /// Changed
32         /// </summary>
33         public bool IsChanged;
34         /// <summary>
35         /// Tray Status
36         /// </summary>
37         public ETrayStatus TrayStatus;
38     }
39
40     /// <summary>
41     /// Tray status information list
42     /// </summary>
43     public List<TrayStatusInfo> TrayStatusInfos = new List<TrayStatusInfo>();
44     }

```

Valid Values for GET_TRAY_STATUS_RESULT.

Enum	Description
CTI_GET_TRAY_STATUS_SUCCESS	successful
CTI_GET_TRAY_STATUS_ERROR	failed
CTI_GET_TRAY_INDEX_ERROR	Tray index error
CTI_GET_TRAY_STATUS_ERROR_NETWORK	Network error
CTI_GET_TRAY_STATUS_ERROR_UNIT	Get unit error
CTI_GET_TRAY_STATUS_ERROR_SPTTMAPPING_INDEX	Sptt mapping index error

abstract void OnEngageTrayFeedBack(ArbinCommandEngageTrayFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostEngageTray(IArbinSocket socket, List<ArbinCommandEngageTrayFeed.EngageTrayInfo> engageTrayList, bool bSubscribe, out int error),

Input:

ArbinCommandEngageTrayFeed cmd;

```
1  public class ArbinCommandEngageTrayFeed : ArbinCommand
2  {
3      public enum ENGAGE_TRAY_RESULT
4      {
5          CTI_ENGAGE_TRAY_SUCCESS,
6          CTI_ENGAGE_TRAY_ERROR,
7          CTI_ENGAGE_TRAY_ERROR_INDEX,
8          CTI_ENGAGE_TRAY_ERROR_NETWORK,
9          CTI_ENGAGE_TRAY_ERROR_UNIT,
10         CTI_ENGAGE_TRAY_ERROR_SPTTMAPPING_INDEX,
11     }
12
13     public class EngageTrayInfo
14     {
15         /// <summary>
16         /// Tray Global Index(starting at 0)
17         /// </summary>
18         public ushort GlobalIndex;
19         /// <summary>
20         /// Error
21         /// </summary>
22         public ENGAGE_TRAY_RESULT Error;
23         /// <summary>
24         /// Changed
25         /// </summary>
26         public bool Engage;
27     }
28
29     /// <summary>
30     /// Engage Tray Info List
31     /// </summary>
32     public List<EngageTrayInfo> EngageTrayInfos = new List<EngageTrayInfo>();
33 }
```

Valid Values for ENGAGE_TRAY_RESULT.

Enum	Description
CTI_ENGAGE_TRAY_SUCCESS,	successful
CTI_ENGAGE_TRAY_ERROR	failed
CTI_ENGAGE_TRAY_ERROR_INDEX	Tray index error
CTI_ENGAGE_TRAY_ERROR_NETWORK	Network error
CTI_ENGAGE_TRAY_ERROR_UNIT	Get unit error
CTI_ENGAGE_TRAY_ERROR_SPTTMAPPING_INDEX	Sptt mapping index error

abstract void OnStartChannelFeedBack (ArbinCommandStartChannelFeed cmd)

- ✓ The callback function for the message returned by the server after sending the bool PostStartChannel (IArbinSocket socket, string TestName, List<ushort> Channels); Check the result stored in cmd to determine the status of the channels.

Input:

ArbinCommandStartChannelFeed cmd;

```
1 public class ArbinCommandStartChannelFeed : ArbinCommand
2 {
3     public enum START_TOKEN
4     {
5         CTI_START_SUCCESS,
6         CTI_START_INDEX = 0x10,
7         CTI_START_ERROR,
8         CTI_START_CHANNEL_RUNNING,
9         CTI_START_CHANNEL_NOT_CONNECT,
10        CTI_START_SCHEDULE_VALID,
11        CTI_START_NO_SCHEDULE_ASSIGNED,
12        CTI_START_SCHEDULE_VERSION,
13        CTI_START_POWER_PROTECTED,
14        CTI_START_RESULTS_FILE_SIZE_LIMIT,
15        CTI_START_STEP_NUMBER,
16        CTI_START_NO_CAN_CONFIGURATON_ASSIGNED,
17        CTI_START_AUX_CHANNEL_MAP,
18        CTI_START_BUILD_AUX_COUNT,
19        CTI_START_POWER_CLAMP_CHECK,
20        CTI_START_AI,
21        CTI_START_SAFOR_GROUPCHAN,
22        CTI_START_BT6000RUNNINGGROUP,
23        CTI_START_CHANNEL_DOWNLOADING_SCHEDULE,
24        CTI_START_DATABASE_QUERY_TEST_NAME_ERROR,
25        CTI_START_TEXTNAME_EXITS,
26        CTI_START_GO_STEP,
27        CTI_START_INVALID_PARALLEL,
28        CTI_START_SAFETY,
29        CTI_START_SECHEDULE_NAME_DIFFERENT,
30        CTI_START_BATTERYSIMULATION_NOT_PARALLEL,
31        CTI_START_CSV_WAIT_TIME,
32        CTI_START_CHANNEL_SUSPENT,
33        CTI_START_TESTNAME_TOO_LONG,
34    };
```



```

35     public START_TOKEN Result;
36 }

```

Valid Values for START_TOKEN Result

Enum	Description
CTI_START_SUCCESS,	Channel started successfully
CTI_START_INDEX = 0x10,	Channel Index error
CTI_START_ERROR,	CTI not allowed to control
CTI_START_CHANNEL_RUNNING,	Channel already running
CTI_START_CHANNEL_NOT_CONNECT,	Channel not connected to server
CTI_START_SCHEDULE_VALID,	Assigned schedule invalid/MUID inconsistent
CTI_START_NO_SCHEDULE_ASSIGNED,	Schedule not assigned
CTI_START_SCHEDULE_VERSION,	Schedule version not consistent
CTI_START_POWER_PROTECTED,	Power protectd
CTI_START_RESULTS_FILE_SIZE_LIMIT,	The RESULTS file is too large to open
CTI_START_STEP_NUMBER,	Step number Error
CTI_START_NO_CAN_CONFIGURATON_ASSIGNED,	Not can configuration assign error
CTI_START_AUX_CHANNEL_MAP,	AUX channel Map error
CTI_START_BUILD_AUX_COUNT,	In Schedule file AUX channel Number error
CTI_START_POWER_CLAMP_CHECK,	PowerClamp the difference between the highest and the lowest values is zero.(1e-8)
CTI_START_AI,	In Schedule file AUX AI Map IV error
CTI_START_SAFOR_GROUPCHAN,	Safor Groupchan
CTI_START_BT6000RUNNINGGROUP,	Bt6000 running group
CTI_START_CHANNEL_DOWNLOADING_SCHEDULE,	Downloading schedule
CTI_START_DATABASE_QUERY_TEST_NAME_ERROR,	Database query text name error
CTI_START_TEXTNAME_EXITS,	text name error
CTI_START_GO_STEP,	step error
CTI_START_INVALID_PARALLEL,	Parallel error
CTI_START_SAFETY,	safety error
CTI_START_SECHEDULE_NAME_DIFFERENT,	schedule name different error
CTI_START_BATTERYSIMULATION_NOT_PARALLEL,	Battery simulation not parallel error
CTI_START_CSV_WAIT_TIME,	Wait 45s until CSV file finished writting
CTI_START_CHANNEL_SUSPENT	Channel suspent

CTI_START_TESTNAME_TOO_LONG

Testname too long

abstract void OnStartChannelAdvancedFeedBack (ArbinCommandStartChannelAdvancedFeed cmd)

- ✓ The callback function for the message returned by the server after sending the bool PostStartAdvancedChannel (IArbinSocket socket, StartChannelAdvancedArgs args).

Input:

ArbinCommandStartChannelAdvancedFeed cmd;

```
1 public class ArbinCommandStartChannelAdvancedFeed : ArbinCommand
2
3 {
4     public class StartChannelResult
5     {
6         public int ChannelIndex { get; set; } = -1;
7         public ArbinCommandStartChannelFeed.START_TOKEN StartResult { get; set; } =
ArbinCommandStartChannelFeed.START_TOKEN.CTI_START_INDEX;
8         public string Message { get; set; } = string.Empty;
9     }
10
11     public long TaskID { get; set; } = 0;
12     public List<int> SuccessfulChannelIDs = new List<int>();
13     public List<StartChannelResult> FailedResults = new List<StartChannelResult>();
14 }
```

Valid Values for START_TOKEN Result

Enum	Description
CTI_START_SUCCESS,	Channel started successfully
CTI_START_INDEX = 0x10,	Channel Index error
CTI_START_ERROR,	CTI not allowed to control
CTI_START_CHANNEL_RUNNING,	Channel already running
CTI_START_CHANNEL_NOT_CONNECT,	Channel not connected to server
CTI_START_SCHEDULE_VALID,	Assigned schedule invalid/MUID inconsistent
CTI_START_NO_SCHEDULE_ASSIGNED,	Schedule not assigned
CTI_START_SCHEDULE_VERSION,	Schedule version not consistent
CTI_START_POWER_PROTECTED,	Power protectd
CTI_START_RESULTS_FILE_SIZE_LIMIT,	The RESULTS file is too large to open
CTI_START_STEP_NUMBER,	Step number Error
CTI_START_NO_CAN_CONFIGURATON_ASSIGNED,	Not can configuration assign error
CTI_START_AUX_CHANNEL_MAP,	AUX channel Map error
CTI_START_BUILD_AUX_COUNT,	In Schedule file AUX channel Number error

CTI_START_POWER_CLAMP_CHECK,	PowerClamp the difference between the highest and the lowest values is zero.(1e-8)
CTI_START_AI,	In Schedule file AUX AI Map IV error
CTI_START_SAFOR_GROUPCHAN,	Safor Groupchan
CTI_START_BT6000RUNNINGGROUP,	Bt6000 running group
CTI_START_CHANNEL_DOWNLOADING_SCHEDULE,	Downloading schedule
CTI_START_DATABASE_QUERY_TEST_NAME_ERROR,	Database query text name error
CTI_START_TEXTNAME_EXITS,	text name error
CTI_START_GO_STEP,	step error
CTI_START_INVALID_PARALLEL,	Parallel error
CTI_START_SAFETY,	safety error
CTI_START_SECHEDULE_NAME_DIFFERENT,	schedule name different error
CTI_START_BATTERYSIMULATION_NOT_PARALLEL,	Battery simulation not parallel error
CTI_START_CSV_WAIT_TIME,	Wait 45s until CSV file finished writting
CTI_START_CHANNEL_SUSPENT	Channel suspent
CTI_START_TESTNAME_TOO_LONG	Testname too long

abstract void OnStopChannelFeedBack (ArbinCommandStopChannelFeed cmd)

- ✓ The callback for handling the message sent by the server in response to bool PostStopChannel (IArbinSocket socket, int nChannel, bool bStopAll);

Input:

ArbinCommandStopChannelFeed cmd;

```

1  public class ArbinCommandStopChannelFeed : ArbinCommand
2      {
3          public enum STOP_TOKEN
4          {
5              SUCCESS = 0,
6              STOP_INDEX = 0x10,
7              STOP_ERROR,
8              STOP_NOT_RUNNING,
9              STOP_CHANNEL_NOT_CONNECT
10         };
11         public STOP_TOKEN Result;
12     }

```

Valid Values for STOP_TOKEN Result.

Enum	Description
SUCCESS	Channel stopped successfully
STOP_INDEX	Channel Index Error

STOP_ERROR	CTI not allowed to control
STOP_NOT_RUNNING	Channel not running
STOP_CHANNEL_NOT_CONNECT	Channel not connected to server

abstract void OnJumpChannelFeedBack (ArbinCommandJumpChannelFeed cmd)

- ✓ The callback function for handling the message returned by the server in response to bool PostJumpChannel (IArbinSocket socket, int stepNum, int channelIndex);

Input:

ArbinCommandJumpChannelFeed cmd;

```

1  public class ArbinCommandJumpChannelFeed : ArbinCommand
2  {
3      public enum JUMP_TOKEN
4      {
5          CTI_JUMP_SUCCESS,
6          CTI_JUMP_INDEX = 0x10,
7          CTI_JUMP_ERROR,
8          CTI_JUMP_CHANNEL_RUNNING,
9          CTI_JUMP_CHANNEL_NOT_CONNECT,
10         CTI_JUMP_SCHEDULE_VALID,
11         CTI_JUMP_NO_SCHEDULE_ASSIGNED,
12         CTI_JUMP_SCHEDULE_VERSION,
13         CTI_JUMP_POWER_PROTECTED,
14         CTI_JUMP_RESULTS_FILE_SIZE_LIMIT,
15         CTI_JUMP_STEP_NUMBER,
16         CTI_JUMP_NO_CAN_CONFIGURATON_ASSIGNED,
17         CTI_JUMP_AUX_CHANNEL_MAP,
18         CTI_JUMP_BUILD_AUX_COUNT,
19         CTI_JUMP_POWER_CLAMP_CHECK,
20         CTI_JUMP_AI,
21         CTI_JUMP_SAFOR_GROUPCHAN,
22         CTI_JUMP_BT6000RUNNINGGROUP,
23         CTI_JUMP_CHANNEL_DOWNLOADING_SCHEDULE,
24         CTI_JUMP_DATABASE_QUERY_TEST_NAME_ERROR,
25         CTI_JUMP_TEXTNAME_EXITS,
26         CTI_JUMP_GO_STEP,
27         CTI_JUMP_INVALID_PARALLEL,
28         CTI_JUMP_SECHEDULE_NAME_DIFFERENT,
29         CTI_JUMP_BATTERYSIMULATION_NOT_PARALLEL,
30         CTI_JUMP_CHANNEL_SUSPENT,
31     };
32     public JUMP_TOKEN Result;
33     public int errorChannel;
34 }

```

Valid Values for JUMP_TOKEN Result.

Enum	Description
CTI_JUMP_SUCCESS	Jump step successful
CTI_JUMP_INDEX	Channel Index Error

CTI_JUMP_ERROR	CTI not allowed to control
CTI_JUMP_CHANNEL_RUNNING	Channel not running
CTI_JUMP_CHANNEL_NOT_CONNECT	Channel not connected to server
CTI_JUMP_SCHEDULE_VALID	Assigned schedule invalid
CTI_JUMP_NO_SCHEDULE_ASSIGNED	Schedule not assigned
CTI_JUMP_SCHEDULE_VERSION	Schedule version
CTI_JUMP_POWER_PROTECTED	Power protectd
CTI_JUMP_RESULTS_FILE_SIZE_LIMIT	The RESULTS file is too large to open
CTI_JUMP_STEP_NUMBER	Step number error
CTI_JUMP_NO_CAN_CONFIGURATON_ASSIGNED	Not can configuration assign error
CTI_JUMP_AUX_CHANNEL_MAP	AUX channel Map error
CTI_JUMP_BUILD_AUX_COUNT	In Schedule file AUX channel Number error
CTI_JUMP_POWER_CLAMP_CHECK	PowerClamp the difference between the highest and the lowest values is zero.(1e-8)
CTI_JUMP_AI	In Schedule file AUX AI Map IV error
CTI_JUMP_SAFOR_GROUPCHAN	Safor Groupchan
CTI_JUMP_BT6000RUNNINGGROUP	Bt6000 running group
CTI_JUMP_CHANNEL_DOWNLOADING_SCHEDULE	Downloading schedule
CTI_JUMP_DATABASE_QUERY_TEST_NAME_ERROR	Database query text name error
CTI_JUMP_TEXTNAME_EXITS	text name error
CTI_JUMP_GO_STEP	step error
CTI_JUMP_INVALID_PARALLEL	Parallel error
CTI_JUMP_SAFETY	Safety
CTI_JUMP_SECHEDULE_NAME_DIFFERENT	Sechedule name different
CTI_JUMP_BATTERYSIMULATION_NOT_PARALLEL	Battery simulation not parallel
CTI_JUMP_CHANNEL_SUSPENT	Channel suspent

abstract void OnResumeChannelFeedBack (ArbinCommandResumChannele Feed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostResumeChannel (IArbinSocket socket, bool AllResume, int Channel Index); Evaluate cmd to find specific information on the status resume command.

Input:

ArbinCommandResumeChannelFeed cmd;

```

1 public class ArbinCommandResumChanneleFeed : ArbinCommand
2 {
```

```

3      public enum RESUME_TOKEN
4      {
5          RESUME_SUCCESS,
6          RESUME_INDEX = 0x10,
7          RESUME_ERROR,
8          RESUME_CHANNEL_RUNNING,
9          RESUME_CHANNEL_NOT_CONNECT,
10         RESUME_SCHEDULE_VALID,
11         RESUME_NO_SCHEDULE_ASSIGNED,
12         RESUME_SCHEDULE_VERSION,
13         RESUME_POWER_PROTECTED,
14         RESUME_RESULTS_FILE_SIZE_LIMIT,
15         RESUME_STEP_NUMBER,
16         RESUME_NO_CAN_CONFIGURATON_ASSIGNED,
17         RESUME_AUX_CHANNEL_MAP,
18         RESUME_BUILD_AUX_COUNT,
19         RESUME_POWER_CLAMP_CHECK,
20         RESUME_AI,
21         RESUME_SAFOR_GROUPCHAN,
22         RESUME_BT0000RUNNINGGROUP,
23         RESUME_CHANNEL_DOWNLOADING_SCHEDULE,
24         RESUME_DATABASE_QUERY_TEST_NAME_ERROR,
25         RESUME_NO_TEST_NAME,
26         RESUME_LOAD_RESUME,
27         RESUME_MAX_MULTIPLE_RESULT,
28         CTI_RESUME_SAFETY,
29         CTI_RESUME_BATTERYSIMULATION_NOT_PARALLEL,
30         CTI_RESUME_CHANNEL_SUSPENT
31     },
32     public RESUME_TOKEN Result;
33 }

```

Valid Values for RESUME_TOKEN Result.

Enum	Description
RESUME_SUCCESS	Test resume successful
RESUME_INDEX	Channel index error
RESUME_ERROR	CTI not allowed to control
RESUME_CHANNEL_RUNNING	Channel already running
RESUME_CHANNEL_NOT_CONNECT	Channel not connected to server
RESUME_SCHEDULE_VALID	Assigned schedule invalid/MUID inconsistent
RESUME_NO_SCHEDULE_ASSIGNED	Schedule not assigned
RESUME_SCHEDULE_VERSION	Schedule version
RESUME_POWER_PROTECTED	Power protected
RESUME_RESULTS_FILE_SIZE_LIMIT	The RESULTS file is too large to open
RESUME_STEP_NUMBER	Step number error
RESUME_NO_CAN_CONFIGURATON_ASSIGNED	Not can configuration assign error
RESUME_AUX_CHANNEL_MAP	AUX channel Map error

RESUME_BUILD_AUX_COUNT	In Schedule file AUX channel Number error
RESUME_POWER_CLAMP_CHECK	PowerClamp the difference between the highest and the lowest values is zero.(1e-8)
RESUME_AI	In Schedule file AUX AI Map IV error
RESUME_SAFOR_GROUPCHAN	Safor Groupchan
RESUME_BT6000RUNNINGGROUP	Bt6000 running group
RESUME_CHANNEL_DOWNLOADING_SCHEDULE	Downloading schedule
RESUME_DATABASE_QUERY_TEST_NAME_ERROR	Database query text name error
RESUME_NO_TEST_NAME	Database query text name is empty
RESUME_LOAD_RESUME	Database query Resume information error
RESUME_MAX_MULTIPLE_RESULT	TestName too many names(Limit 128)
CTI_RESUME_SAFETY	Safety
CTI_RESUME_BATTERYSIMULATION_NOT_PARALLEL	Battery simulation not parallel
CTI_RESUME_CHANNEL_SUSPENT	Channel suspend

abstract void OnContinueChannelFeedBack(ArbinCommandContinueChannelFeed cmd)

- ✓ The callback function for handling messages sent from the server following a bool PostContinueChannel(IArbinSocket socket, List<ushort> Channels) Evaluate cmd to find specific information on the status continue command.

Input:

ArbinCommandContinueChannelFeed cmd;

```

1 public class ArbinCommandContinueChannelFeed: ArbinCommand
2 {
3     public enum CONTINUE_TOKEN
4     {
5         CTI_CONTINUE_SUCCESS,
6         CTI_CONTINUE_INDEX = 0x10,
7         CTI_CONTINUE_ERROR,
8         CTI_CONTINUE_CHANNEL_RUNNING,
9         CTI_CONTINUE_CHANNEL_NOT_CONNECT,
10        CTI_CONTINUE_CHANNEL_CALIBRATING,
11        CTI_CONTINUE_NOT_PAUSE_NORMAL,
12        CTI_CONTINUE_CHANNEL_UNSAFE
13    },
14    public CONTINUE_TOKEN Result;
15 }
```

Valid Values for CONTINUE_TOKEN Result.

Enum	Description
CTI_CONTINUE_SUCCESS	OK
CTI_CONTINUE_INDEX	Channel index error

CTI_CONTINUE_ERROR	Execute error
CTI_CONTINUE_CHANNEL_RUNNING	Channel already running
CTI_CONTINUE_CHANNEL_NOT_CONNECT	Channel not connected to server
CTI_CONTINUE_CHANNEL_CALIBRATING	Channel Calibrating
CTI_CONTINUE_NOT_PAUSE_NORMAL	Not Pause Normal
CTI_CONTINUE_CHANNEL_UNSAFE	Channel Unsafe

abstract void OnBrowseDirectoryBack (ArbinCommandBrowseDirectoryFeed cmd)

- ✓ The callback for handling the message from the cycler in response to bool PostBrowseDirectory (IArbinSocket socket, string strPath); Use this to process the information gathered from the server.

Input:

ArbinCommandBrowseDirectoryFeed cmd;

```

1 public class ArbinCommandBrowseDirectoryFeed : ArbinCommand
2 {
3     public enum BROWSE_DIRECTORY_RESULT
4     {
5         CTI_BROWSE_DIRECTORY_SUCCESS = 1,
6         CTI_BROWSE_SCHEDULE_SUCCESS,
7         CTI_BROWSE_SCHEDULE_VERSION1_SUCCESS,
8         CTI_BROWSE_DIRECTORY_FAILED,
9     };
10    public struct DirFileInfo
11    {
12        public uint Type;
13        public string DirFileName;
14        public int dwSize;
15        public string wcModified;
16    };
17    public BROWSE_DIRECTORY_RESULT Result;
18    public List<DirFileInfo> DirFileInfoList = null;
19 }
```

Name	DataType	Description
BROWSE_DIRECTORY_RESULT Result	Enum	Result contains the success message from the cycler
CTI_BROWSE_DIRECTORY_SUCCESS		Directory found and accessed
CTI_BROWSE_SCHEDULE_SUCCESS		Successfully obtained the Schedule file using the "SCHEDULE" keyword
CTI_BROWSE_SCHEDULE_VERSION1_SUCCESS		Successfully obtained the Schedule file using the "SCHEDULE" keyword, Use ArbinStruct.ScheduleFileRelativePath structure

CTI_BROWSE_DIRECTORY_FAILED		This path is not part of the MITS_PRO exclusive folder
DirFileInfoList	List<DirFileInfo>	list containing information on everything listed under the requested directory, including other directories.
Type	uint	0 - Directory 1 - File
DirFileName	string	Full path of parent directory of a file
dwSize	int	Length of file in bytes
wcModified	string	Time when the file/directory was last written

abstract void OnDeleteFolderBack(ArbinCommandDeleteFolderFeed cmd)

- ✓ The callback function for handling the message returned by the server in response to bool PostDeleteFolder (IArbinSocket socket, string strPath)

Input:

ArbinCommandDeleteFolderFeed cmd;

```

1 public class ArbinCommandDeleteFolderFeed : ArbinCommand
2 {
3     /// <summary>
4     /// Result Type
5     /// </summary>
6     public enum RESULT_TYPE
7     {
8         CTI_NEW_SUCCESS = 1,
9         CTI_DELETE_SUCCESS,
10        CTI_NEW_FAILED,
11        CTI_NEW_FAILED_ADD_FOLDER,
12        CTI_DELETE_FAILED,
13        CTI_DELETE_FAILED_EXTENSION,
14        CTI_DELETE_FAILED_TEXT_RUNNING,
15        CTI_DELETE_FAILED_EXIST
16    };
17
18    public RESULT_TYPE Result;    //(RESULT_TYPE) Return result
19 }
```

Valid Values For RESULT_TYPE Result

Enum	Description
CTI_NEW_SUCCESS	Not to be used.
CTI_DELETE_SUCCESS	File deleted successfully

CTI_NEW_FAILED	Not to be used
CTI_NEW_FAILED_ADD_FOLDER	Not to be used.
CTI_DELETE_FAILED	File not deleted
CTI_DELETE_FAILED_EXTENSION	Files with this extension are not allowed to be deleted
CTI_DELETE_FAILED_TEXT_RUNNING	Test is running on schedule file attempted to delete
CTI_DELETE_FAILED_EXIST	File does not exist

abstract void OnDownloadFileBack(ArbinCommandDownloadFileFeed cmd)

- ✓ The callback function for handling the server's response to bool PostDownloadFile (IArbinSocket socket, string strPath, double time); Use this to handle the file data sent by the server.

Input:

ArbinCommandDownloadFileFeed cmd;

```

1 public class ArbinCommandDownloadFileFeed : ArbinCommand
2 {
3     public enum DOWNLOAD_RESULT
4     {
5         CTI_DOWNLOAD_SUCCESS = 1,
6         CTI_DOWNLOAD_FAILED,
7         CTI_DOWNLOAD_MD5_ERR,
8         CTI_DOWNLOAD_MAX_LENGTH_ERR
9     };
10    public DOWNLOAD_RESULT Result;
11    public string m_MD5;
12    public ulong dwFileLength;
13    public byte[] byData = null;
14    public double DownloadTime;
15    public double UploadTime;
16    public uint uGeneralPackage;
17    public uint uPackageIndex;
18 }
```

Valid Vales For DOWNLOAD_RESULT Result

Enum	Data Type	Description
DOWNLOAD_RESULT Result	Enum	Result contains the success message from the cyclor
CTI_DOWNLOAD_SUCCESS		File downloaded successfully
CTI_DOWNLOAD_FAILED		File download failed
CTI_DOWNLOAD_MD5_ERR		MD5 hash does not match
CTI_DOWNLOAD_MAX_LENGTH_ERR		File size exceeded
m_MD5	string	m_MD5 contains the MD5 checksum value of the chunk. Compare this with the computed MD5 for the chunk to ensure correct data

		transmission. If the MD5 values do not match, reject the transmission, and try again.
dwFileLength	ulong	File length in bytes
byData	byte[]	byData contains the data for this chunk
DownloadTime	double	Time to start download
UploadTime	double	Time to start upload
uGeneralPackage	unit	uGeneralPackage is the number of packages being sent to construct the file
uPackageIndex	unit	uPackageIndex is the index of the current chunk within the file

i The file data is sent in chunks. cmd contains information on the chunk index and the number of chunks being sent. cmd allows for checking the validity of the data sent by the cycler. This callback must be implemented to save the file to the CTI computer.

abstract void OnNewOrDeleteBack(ArbinCommandNewOrDeleteFeed cmd)

- ✓ The callback function for handling the message returned by the server in response to bool PostNewOrDelete(IArbinSocket socket, string strPath, ArbinCommandNewOrDeleteFeed.NEW_OR_DELETE_TYPE uType)

Input:

ArbinCommandNewOrDeleteFeed cmd;

```

1 public class ArbinCommandNewOrDeleteFeed : ArbinCommand
2 {
3     public enum NEW_OR_DELETE_RESULT
4     {
5         CTI_NEW_SUCCESS = 1,
6         CTI_DELETE_SUCCESS,
7         CTI_NEW_FAILED,
8         CTI_NEW_FAILED_ADD_FOLDER,
9         CTI_DELETE_FAILED,
10        CTI_DELETE_FAILED_EXTENSION,
11        CTI_DELETE_FAILED_TEXT_RUNNING,
12        CTI_DELETE_FAILED_EXIST
13    };
14
15    public enum NEW_OR_DELETE_TYPE
16    {
17        CTI_DELETE = 0,
18        CTI_NEW = 1
19    };
20
21    public NEW_OR_DELETE_RESULT Result;
22 }
```

Valid Values For NEW_OR_DELETE_RESULT Result

Enum	Description
CTI_NEW_SUCCESS	New Folder created successfully
CTI_DELETE_SUCCESS	
CTI_NEW_FAILED	Failed to create Folder
CTI_NEW_FAILED_ADD_FOLDER	This directory cannot add folders
CTI_DELETE_FAILED	
CTI_DELETE_FAILED_EXTENSION	
CTI_DELETE_FAILED_TEXT_RUNNING	
CTI_DELETE_FAILED_EXIST	

abstract void OnNewFolderBack(ArbinCommandNewFolderFeed cmd)

✓ The callback function for handling the message returned by the server in response to bool PostNewFolder (IArbinSocket socket, string strPath);

Input:

ArbinCommandNewFolderFeed cmd;

```
1 public class ArbinCommandNewFolderFeed : ArbinCommand
2 {
3     public enum RESULT_TYPE
4     {
5         CTI_NEW_SUCCESS = 1,
6         CTI_DELETE_SUCCESS,
7         CTI_NEW_FAILED,
8         CTI_NEW_FAILED_ADD_FOLDER,
9         CTI_DELETE_FAILED,
10        CTI_DELETE_FAILED_EXTENSION,
11        CTI_DELETE_FAILED_TEXT_RUNNING,
12        CTI_DELETE_FAILED_EXIST
13    };
14
15    public RESULT_TYPE Result;
16 }
```

Valid Values For RESULT_TYPE Result

Enum	Description
CTI_NEW_SUCCESS	New Folder created successfully
CTI_DELETE_SUCCESS	Not to be used.
CTI_NEW_FAILED	Failed to create Folder
CTI_NEW_FAILED_ADD_FOLDER	This directory cannot add folders
CTI_DELETE_FAILED	Not to be used.
CTI_DELETE_FAILED_EXTENSION	Not to be used.

CTI_DELETE_FAILED_TEXT_RUNNING	Not to be used.
CTI_DELETE_FAILED_EXIST	Not to be used.

abstract void OnUploadFileBack(ArbinCommandUploadFileFeed cmd)

- ✓ The callback for the message sent from the server in response to bool PostUploadFile (IArbinSocket socket, string strPath, byte[] Filedata, double time, uint uGeneralPackage, uint PackageIndex); Use this to check the success or failure of the file upload command.

Input:

ArbinCommandUploadFileFeed cmd;

```

1 public class ArbinCommandUploadFileFeed : ArbinCommand
2 {
3     public enum UPLOAD_RESULT
4     {
5         CTI_UPLOAD_SUCCESS = 1,
6         CTI_UPLOAD_FAILED,
7         CTI_UPLOAD_MD5_ERR,
8         CTI_UPLOAD_FAILED_TEXT_RUNNING
9         CTI_UPLOAD_FILE_EXIST_WITH_DIFFERENT_MD5,
10        CTI_UPLOAD_FILE_EXIST_WITH_SAME_MD5,
11        CTI_UPLOAD_FILE_EXIST_NOT_OVERRIDE,
12        CTI_UPLOAD_FAILED_USER_CANCEL,
13        CTI_UPLOAD_FAILED_TIMEOUT,
14        CTI_UPLOAD_FAILED_CHECK_FILE_TIMEOUT,
15        CTI_UPLOAD_IN_PROGRESS,
16    };
17    public UPLOAD_RESULT Result;
18    public double UploadTime;
19    public uint uGeneralPackage;
20    public uint uPackageIndex;
21    public class CUploadFileResult
22    {
23        /// Refers to the method to be called on the corresponding asynchronous operation
24        public delegate void AsyncCallback(ref CUploadFileResult uploadFileResult);
25        public UPLOAD_RESULT ResultCode = UPLOAD_RESULT.CTI_UPLOAD_IN_PROGRESS;
26        public bool IsCancelUploadFile = false;
27        public float ProgressRate = 0.0f;
28    }
29 }
```

Valid Values For UPLOAD_RESULT Result

Enum	Data Type	Description
UPLOAD_RESULT Result	Enum	Result contains the success message from the cyclor
CTI_UPLOAD_SUCCESS		File uploaded successfully
CTI_UPLOAD_FAILED		File upload failed
CTI_UPLOAD_MD5_ERR		MD5 hash does not match
CTI_UPLOAD_FAILED_TEXT_RUNNING		System running the test

UploadTime	double	Time to start upload
uGeneralPackage	unit	uGeneralPackage is the number of packages being sent to construct the file
uPackageIndex	unit	uPackageIndex is the index of the current chunk within the file
CUploadFileResult	Class	method to be called on the corresponding asynchronous operation

abstract void OnCheckFileExBack(ArbinCommandCheckFileExFeed cmd);

- ✓ The callback for the message sent from the server in response to bool PostCheckFileEx(IArbinSocket socket, string strRemoteRelativeFile Name, byte[] MD5); Use this to check the existence/integrity of the file on the server.

Input:

ArbinCommandCheckFileExFeed cmd;

```

1 public class ArbinCommandCheckFileExFeed : ArbinCommand
2 {
3     public bool bRelativePathExist;
4     public bool bFileNameExist;
5     public bool bMD5Exist;
6     public string FilePath;
7     public string Reason;
8 }
```

abstract void OnAssignFileFeedBack(ArbinCommandAssignFileFeed cmd);

- ✓ The callback for the message sent from the server in response to bool PostAssignFile(IArbinSocket socket, string fileName, bool bAssignAll, ArbinCommandAssignFileFeed.EFileKind eFileKind, List<ushort> chanList) Use this to assign schedule, CANBMS, SMB files to a channel.

Input:

ArbinCommandAssignFileFeed cmd;

```

1 public class ArbinCommandAssignFileFeed : ArbinCommand
2 {
3     public enum EFileKind
4     {
5         None = -1,
6         Schedule,
7         CANBMS,
8         SMB,
9         Simulation,
10        BatterySimulation,
11        TestObject,
12        Chart,
13        BaseFileCount = Chart,
14        FileKindCount,
15    }
16     public enum ASSIGN_TOKEN
```


```

17     {
18         CTI_ASSIGN_SUCCESS,
19         CTI_ASSIGN_FAILED,
20         CTI_ASSIGN_INDEX = 0x10,
21         CTI_ASSIGN_ERROR,
22         CTI_ASSIGN_FILE_NAME_EMPTY_ERROR,
23         CTI_ASSIGN_FILE_NOT_FIND_ERROR,
24         CTI_ASSIGN_CHANNEL_RUNNING_ERROR,
25         CTI_ASSIGN_CHANNEL_DOWNLOAD_ERROR,
26         CTI_ASSIGN_BACTH_FILE_OPENED,
27         CTI_ASSIGN_FILE_CANNOT_ASSIGN,
28         CTI_ASSIGN_FILE_SAVE_FAILED,
29         CTI_ASSIGN_FILE_UNSUPPORTED_FILE_TYPE,
30         CTI_ASSIGN_FILE_NOT_ASSIGN_SCHEDULE,
31         CTI_ASSIGN_FILE_SCHEDULE_NOT_AUX_REQUIREMENT,
32         CTI_ASSIGN_FILE_SCHEDULE_IS_RUNNING,
33         CTI_ASSIGN_SCHEDULE_MUID_NOT_SAME,
34         CTI_ASSIGN_FILE_CLEAR
35     };
36     public ASSIGN_TOKEN Result;
37     public SortedDictionary< ASSIGN_TOKEN, List<int>> ChanListResultPairs;
38     public string Reason;
39 }

```

Valid Values for ASSIGN_TOKEN Result.

Enum	Description
CTI_ASSIGN_SUCCESS,	File assign Successful
CTI_ASSIGN_FAILED,	File assign fail
CTI_ASSIGN_INDEX = 0x10,	Channel Index error
CTI_ASSIGN_ERROR,	CTI not allowed to control
CTI_ASSIGN_FILE_NAME_EMPTY_ERROR,	File name is empty
CTI_ASSIGN_FILE_NOT_FIND_ERROR,	File not found in work directory of server
CTI_ASSIGN_CHANNEL_RUNNING_ERROR,	Channel already running
CTI_ASSIGN_CHANNEL_DOWNLOAD_ERROR,	Downloading file
CTI_ASSIGN_BACTH_FILE_OPENED,	Batch file was opened
CTI_ASSIGN_FILE_CANNOT_ASSIGN,	assign error
CTI_ASSIGN_FILE_SAVE_FAILED,	Failed save
CTI_ASSIGN_FILE_UNSUPPORTED_FILE_TYPE,	Unsupported file type
CTI_ASSIGN_FILE_NOT_ASSIGN_SCHEDULE,	Not assign schedule
CTI_ASSIGN_FILE_SCHEDULE_NOT_AUX_REQUIREMENT,	Schedule not aux requirement
CTI_ASSIGN_FILE_SCHEDULE_IS_RUNNING,	Schedule is running
CTI_ASSIGN_SCHEDULE_MUID_NOT_SAME,	MUID of schedule file not consistent
CTI_ASSIGN_FILE_CLEAR,	Clear file

 If a schedule file is copy/pasted from another machine its MUID will be inconsistent. It either needs to be created locally or uploaded from CTI.

abstract void OnSendMsgToCTIBack (ArbinCommandSendMsgToCTIFeed cmd)

- ✓ The callback function for the message returned by the server after sending the bool PostSendMsgToCTI (IArbinSocket socket, string msg);

Input:

ArbinCommandSendMsgToCTIFeed cmd;

```
1 public class ArbinCommandSendMsgToCTIFeed : ArbinCommand
2     {
3         public enum SEND_MSG_TO_CTI_RESULT
4         {
5             SEND_MSG_TO_CTI_SUCCESS = 1,
6             SEND_MSG_TO_CTI_FAILED
7         };
8         public SEND_MSG_TO_CTI_RESULT Result;
9     }
```

Valid Values For SEND_MSG_TO_CTI_RESULT Result

Enum	Description
SEND_MSG_TO_CTI_SUCCESS	Message Sent
SEND_MSG_TO_CTI_FAILED	Message not sent

abstract OnStartAutomaticCalibrationBack(ArbinCommandStartAutomaticCalibrationFeed cmd)

- ✓ The callback function for the message returned by the server after sending the bool PostStartAutomaticCalibration (IArbinSocket socket);

Input:

ArbinCommandStartAutomaticCalibrationFeed cmd;

```
1 public class ArbinCommandStartAutomaticCalibrationFeed: ArbinCommand
2     {
3         public enum AUTOCALI_START_RESULT
4         {
5             CTI_AUTOCALI_START_SUCCESS = 1,
6             CTI_AUTOCALI_START_FAILED
7         };
8         public AUTOCALI_START_RESULT Result;
9     }
```

Valid Values for AUTOCALI_START_RESULT Result

Enum	Description
CTI_AUTOCALI_START_SUCCESS	Auto Cali Start successful
CTI_AUTOCALI_START_FAILED	Auto Cali Start failed

abstract void OnGetChannelsDataFeedBack (ArbinCommandGetChannelDataFeed cmd)

- ✓ The callback for handling the message received from the server in response to bool PostGetChannelsData (IArbinSocket socket, uint NeedType, short OnlyGet ChannelNumber, ArbinCommandGetChannelFeed.GET_CHANNEL_TYPE GetChannelType); Use this to process the state of the channels requested.

Input:

ArbinCommandGetChannelDataFeed cmd;

```
1 public class ArbinCommandGetChannelDataFeed: ArbinCommand
2 {
3     public enum GET_CHANNELS_INFO_NEED_TYPE
4     {
5         THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_BMS = 0x100,
6         THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_SMB = 0x200,
7         THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_AUX = 0x400,
8     };
9     public enum GET_CHANNEL_TYPE
10    {
11        ALLCHANNEL = 1,
12        RUNNING = 2,
13        UNSAFE = 3
14    };
15    public enum ChannelStatus
16    {
17        Idle,
18        Transition,
19        Charge,
20        Discharge,
21        Rest,
22        Wait,
23        External_Charge,
24        Calibration,
25        Unsafe,
26        Pulse,
27        Internal_Resistance,
28        AC_Impedance,
29        ACI_Cell,
30        Test_Settings,
31        Error,
32        Finished,
33        Volt_Meter,
34        Waiting_for_ACS,
35        Pause,
36        EMPTY,
37        Idle_from_MCU,
38        Start,
39        Runing,
40        Step_Transfer,
41        Resume,
42        Go_Pause,
43        Go_Stop,
44        Go_Next_Step,
45        Online_Update,
46        Daq_Memory_Unsafe,
47        ACR,
48    };
49    public struct AuxData
50    {
```

```

51         public float Value;
52         public float dtValue;
53     }
54     public struct CANInfo
55     {
56         public uint nIndex;
57         public double Value;
58         public string Unit;
59     }
60     public struct SMBInfo
61     {
62         public uint nIndex;
63         public uint nType;
64         public string Unit;
65         public object Value;
66     }
67     public class ChannelInfo
68     {
69         public enum AUX_TYPE
70         {
71             AuxV,
72             T,
73             P,
74             pH,
75             FR,
76             Conc,
77             DI,
78             DO,
79             EC,
80             Safety,
81             Humidity,
82             AO,
83             MAX_NUM,
84         };
85         public static readonly ChannelInfo Empty = new ChannelInfo();
86         public uint Channel = uint.MaxValue;
87         public ChannelStatus Status = ChannelStatus.Idle;
88         public bool CommFailure = true;
89         public string Schedule;
90         public string CANCEfg;
91         public string SMBCfg;
92         public string Testname;
93         public string ExitCondition;
94         public string StepAndCycle;
95         public string Barcode;
96         public uint MasterChannel;
97         public double TestTime;
98         public double StepTime;
99         public float Voltage;
100        public float Current;
101        public float Power;
102        public float ChargeCapacity;
103        public float DischargeCapacity;
104        public float ChargeEnergy;
105        public float DischargeEnergy;
106        public float InternalResistance;
107        public float dvdt;
108        public float ACR;

```

```

109         public float ACI;
110         public float ACIPhase;
111         public List<CANInfo> CAN = null;
112         public List<SMBInfo> SMB = null;
113         public List<AuxData>[] Auxs = new List<AuxData>[(int)AUX_TYPE.MAX_NUM];
114     }
115     public List<ChannelInfo> m_Channels = new List<ChannelInfo>();
116 }


```

Valid Values for GET_CHANNELS_INFO_NEED_TYPE Result and GET_CHANNEL_TYPE

Name	Data Type	Description
GET_CHANNELS_INFO_NEED_TYPE	Enum	contains the information type required
THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_BMS		Returns BMS info of the channels
THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_SMB		Returns SMBinfo of the channels
THIRD_PARTY_GET_CHANNELS_INFO_NEED_TYPE_AUX		Returns AUX info of the channels
GET_CHANNEL_TYPE	Enum	Contains channel information
ALLCHANNEL		Info of all channels
RUNNING		Info of channels running tests
UNSAFE		Info of unsafe channels
AuxData	struct	Auxiliary data and its differentiation of time
CANinfo	struct	CAN messages
SMBinfo	struct	SMB messages
CommFailure	bool	false - no problem with the connection true - can't connect to MCU
ExitCondition	string	Stop condition of test
MasterChannel	uint	Main channel in parallel connection
ACR	float	Measured value of battery internal resistance
ACI	float	Calculated value of impedance resulting from 1kHz imposed sine wave
ACIPhase	float	Phase angle value of the AC impedance in degree

abstract void OnGetChannelsDataSPTTFeedBack (ArbinCommandGetChannelDataSPTTFeed cmd)

- ✓ The callback for handling the message received from the server in response to bool PostGetChannelsDataSPTT(IArbinSocket socket, short OnlyGetChannelNumber = -1); Use this to process the state of the channels requested.

 (For MITS 10, In development)

Input:

ArbinCommandGetChannelDataSPTTFeed cmd;

```
1 public class ArbinCommandGetChannelDataSPTTFeed : ArbinCommand
2 {
3     public enum ChannelDataType
4     {
5         /// <summary>
6         /// EQ
7         /// </summary>
8         EQ = 0,
9         /// <summary>
10        /// CELL
11        /// </summary>
12        CELL = 1,
13    };
14
15    public enum GetChannelDataSPTTResult
16    {
17        /// <summary>
18        /// Success
19        /// </summary>
20        SUCCESS = 0,
21        /// <summary>
22        /// Congfig has no SPTT configured
23        /// </summary>
24        CONFIG_HAS_NO_SPTT = 0x10,
25        /// <summary>
26        /// Channel has no SPTT configured
27        /// </summary>
28        CHANNEL_HAS_NO_SPTT = 1,
29    };
30
31    public class ChannelInfoSPTT : IComparer<ChannelInfoSPTT>
32    {
33        /// <summary>
34        /// Represents the empty value. This field is read-only.
35        /// </summary>
36        public static readonly ChannelInfoSPTT Empty = new ChannelInfoSPTT();
37
38        /// <summary>
39        /// CELL/EQ Global Index starts from zero.
40        /// </summary>
41        public int GlobalIndex;
42        /// <summary>
43        /// Voltage
44        /// </summary>
45        public float Voltage;
46        /// <summary>
47        /// Current
48        /// </summary>
```

```

49     public float Current;
50     /// <summary>
51     /// Power
52     /// </summary>
53     public float Power;
54     /// <summary>
55     /// ChargeCapacity
56     /// </summary>
57     public float ChargeCapacity;
58     /// <summary>
59     /// DischargeCapacity
60     /// </summary>
61     public float DischargeCapacity;
62     /// <summary>
63     /// ChargeEnergy
64     /// </summary>
65     public float ChargeEnergy;
66     /// <summary>
67     /// DishargeEnergy
68     /// </summary>
69     public float DishargeEnergy;
70     /// <summary>
71     /// InternalResistance
72     /// </summary>
73     public float InternalResistance;
74     /// <summary>
75     /// AuxiliaryTemperature
76     /// </summary>
77     public float AuxiliaryTemperature;
78     /// <summary>
79     /// Status
80     /// </summary>
81     public ArbinCommandGetChannelDataFeed.ChannelStatus Status;
82     /// <summary>
83     /// DataType
84     /// </summary>
85     public ChannelDataType DataType;
86
87     /// <summary>
88     /// Compares two objects and returns a value that indicates whether one object is less than,
equal to, or greater than the other object
89     /// </summary>
90     /// <returns></returns>
91     public int Compare(ChannelInfoSPTT x, ChannelInfoSPTT y)
92     {
93         if (ReferenceEquals(x, y))
94         {
95             return 0;
96         }
97
98         if (x != null && y != null)
99             return x.GlobalIndex.CompareTo(y.GlobalIndex);
100         else
101             return 0;
102     }
103 }
104
105 public class EQData : ChannelInfoSPTT

```

```

106     {
107         /// <summary>
108         /// IV channel global Index
109         /// </summary>
110         public int IVChannelIndex = 0;
111
112         /// <summary>
113         /// Cell data list
114         /// </summary>
115         public List<ChannelInfoSPTT> CELLS = new List<ChannelInfoSPTT>();
116     }
117
118     /// <summary>
119     /// Collection of channels
120     /// </summary>
121     /// <seealso cref="EQData"/>
122     public List<EQData> Channels = new List<EQData>();
123 }

```

abstract void OnGetChannelsDataMinimalistModeFeedBack(ArbinCommand GetChannelDataMinimalistModeFeed cmd);

- ✓ The callback for handling the message received from the server in response to bool PostGetChannelsDataMminimalistMode(IArbinSocket socket, short onlyGetChannelNumber = -1) This returns only current and voltage.

Input:

ArbinCommandGetChannelDataMinimalistModeFeed cmd;

```

1  public class ArbinCommandGetChannelDataMinimalistModeFeed : ArbinCommand
2  {
3      public class ChannelInfo
4      {
5          /// <summary>
6          /// Represents the empty value. This field is read-only.
7          /// </summary>
8          public static readonly ChannelInfo Empty = new ChannelInfo();
9          /// <summary>
10         /// Channel number starts from zero.
11         /// </summary>
12         public uint Channel = uint.MaxValue;
13         /// <summary>
14         /// <para>Measured value of present channel voltage.</para>
15         /// </summary>
16         public float Voltage;
17         /// <summary>
18         /// <para>Measured value of present channel current.</para>
19         /// </summary>
20         public float Current;
21     }
22     /// <summary>
23     /// Collection of channels
24     /// </summary>
25     public List<ChannelInfo> m_Channels = new List<ChannelInfo>();
26 }

```

abstract void OnGetChannelsDataSimpleModeFeedBack(ArbinCommand GetChannelDataSimpleModeFeed cmd);

- ✓ The callback for handling the message received from the server in response to bool PostGetChannelsDataSimpleMode(IArbinSocket socket, short OnlyGet ChannelNumber = -1, ArbinCommandGetChannelDataSimpleModeFeed. GET_CHANNEL_TYPE GetChannelType = ArbinCommandGetChannelDataSimpleModeFeed.GET_CHANNEL_TYPE.ALLCHANNEL)

Input:

ArbinCommandGetChannelDataSimpleModeFeed cmd;

```
1 public class ArbinCommandGetChannelDataSimpleModeFeed : ArbinCommand
2 {
3     public enum GET_CHANNEL_TYPE
4     {
5         ALLCHANNEL = 1,
6         RUNNING = 2,
7         UNSAFE = 3
8     };
9
10    public class ChannelInfo
11    {
12        public enum AUX_TYPE
13        {
14            AuxV,
15            T,
16            P,
17            pH,
18            FR,
19            Conc,
20            DI,
21            DO,
22            EC,
23            Safety,
24            Humidity,
25            AO,
26            MAX_NUM,
27        };
28
29        public static readonly ChannelInfo Empty = new ChannelInfo();
30        public uint Channel = uint.MaxValue;
31        public ArbinCommandGetChannelDataFeed.ChannelStatus Status = ArbinCommand
32        GetChannelDataFeed.ChannelStatus.Idle;
33        public bool CommFailure = true;
34        public uint MasterChannel;
35        public double TestTime;
36        public double StepTime;
37        public float Voltage;
38        public float Current;
39        public List<ArbinCommandGetChannelDataFeed.AuxData>[] Auxs = new List<Arbin
40        CommandGetChannelDataFeed.AuxData>[(int)AUX_TYPE.MAX_NUM];
41    }
42    public List<ChannelInfo> m_Channels = new List<ChannelInfo>();
43 }
```

abstract void OnGetResumeDataBack (ArbinCommandGetResumeDataFeed cmd)

- ✓ The callback function for bool PostGetResumeData (IArbinSocket socket, List<ushort> channels); Currently only gives the channel number.

Input:

ArbinCommandGetResumeDataFeed cmd;

```
1 public class ArbinCommandGetResumeDataFeed : ArbinCommand
2 {
3     public class ResumeDataInfo
4     {
5         public static readonly ResumeDataInfo Empty = new ResumeDataInfo();
6
7         public uint Channel = uint.MaxValue;
8
9         public struct RESUME_DATA
10        {
11            public uint Cycle;
12            public double TestTime;
13            public double StepTime;
14            public double CCapacity;
15            public double DCapacity;
16            public double CEnergy;
17            public double DEnergy;
18            public double TC_Time1;
19            public double TC_Time2;
20            public double TC_Time3;
21            public double TC_Time4;
22            public double TC_CCapacity1;
23            public double TC_CCapacity2;
24            public double TC_DCapacity1;
25            public double TC_DCapacity2;
26            public double TC_CEnergy1;
27            public double TC_CEnergy2;
28            public double TC_DEnergy1;
29            public double TC_DEnergy2;
30            public float TC_Counter1;
31            public float TC_Counter2;
32            public float TC_Counter3;
33            public float TC_Counter4;
34            public double ChargeCapacityTime;
35            public double DischargeCapacityTime;
36            public float MVUD1;
37            public float MVUD2;
38            public float MVUD3;
39            public float MVUD4;
40            public float MVUD5;
41            public float MVUD6;
42            public float MVUD7;
43            public float MVUD8;
44            public float MVUD9;
45            public float MVUD10;
46            public float MVUD11;
47            public float MVUD12;
48            public float MVUD13;
49            public float MVUD14;
50            public float MVUD15;
51            public float MVUD16;
52        }
53        public uint channelCode;
54        public string TestName;
55        public string Schedule;
56        public string Createor;
```



```

57         public string Comment;
58         public string StartTime;
59         public RESUME_DATA ResumeData;
60
61         public List<string> Steps = null;
62     }
63     public List<ResumeDataInfo> m_Channels = new List<ResumeDataInfo>();
64 }
65

```

abstract void OnGetSerialNumberFeedBack (ArbinCommandGetSerialNumberFeed cmd);

- ✓ The callback function for handling the message returned by the server in response to bool PostGetSerialNumber (IArbinSocket socket);

Input:

ArbinCommandGetSerialNumberFeed cmd;

```

1  public class ArbinCommandGetSerialNumberFeed : ArbinCommand
2  {
3      public enum ASSIGN_TOKEN
4      {
5          CTI_GET_SERIAL_SUCCESS,
6          CTI_ASSIGN_ERROR = 0x10,
7      };
8      public double SerialNum;
9      public ASSIGN_TOKEN Result;
10 }

```

Valid Values For ASSIGN_TOKEN Result

Enum	Description
CTI_GET_SERIAL_SUCCESS	Serial Number fetched successfully.
CTI_ASSIGN_ERROR	

abstract void OnGetStartDataBack (ArbinCommandGetStartDataFeed cmd)

- ✓ Feedback is returned by the server when a bool PostGetStartData (IArbinSocket socket, List<ushort> channels); command is issued.

Input:

ArbinCommandGetStartDataFeed cmd;

```

1  public class ArbinCommandGetStartDataFeed : ArbinCommand
2  {
3      public class StartDataInfo
4      {
5          public static readonly StartDataInfo Empty = new StartDataInfo();
6          public uint Channel = uint.MaxValue;
7          public uint channelCode;
8          public string Schedule;
9          public float fMV_UD1;
10         public float fMV_UD2;
11         public float fMV_UD3;
12         public float fMV_UD4;
13         public float fMV_UD5;

```

```

14         public float fMV_UD6;
15         public float fMV_UD7;
16         public float fMV_UD8;
17         public float fMV_UD9;
18         public float fMV_UD10;
19         public float fMV_UD11;
20         public float fMV_UD12;
21         public float fMV_UD13;
22         public float fMV_UD14;
23         public float fMV_UD15;
24         public float fMV_UD16;
25         public List<string> TestNames = null;
26         public List<string> Steps = null;
27     }
28     public List<StartDataInfo> m_Channels = new List<StartDataInfo>();
29 }

```

abstract void

OnGetServerSoftwareVersionNumberFeedBack(ArbinCommandGetServerSoftwareVersionNumberFeed cmd)

- ✓ The callback function for the server's response to bool PostGetServerSoftwareVersionNumber(IArbinSocket socket); Used to determine if the version number of the Cycler was successfully obtained.

Input:

ArbinCommandGetServerSoftwareVersionNumberFeed cmd;

```

1 public class ArbinCommandGetServerSoftwareVersionNumberFeed : ArbinCommand
2 {
3     /// <summary>
4     /// Server software version number
5     /// </summary>
6     public string ServerVersionNumber = string.Empty;
7 }

```

abstract void OnGetStringLimitLengthFeedBack(ArbinCommandGetStringLimitLengthFeed cmd)

- ✓ The callback function for the server's response to bool PostGetStringLimitLength(IArbinSocket socket, uint NeedType); Used to determine if the maximum length of the string was successfully obtained.

Input:

ArbinCommandGetStringLimitLengthFeed cmd;

```

1 public class ArbinCommandGetStringLimitLengthFeed : ArbinCommand
2 {
3     /// <summary>
4     /// String limit length type
5     /// </summary>
6     public enum ECTIStringLimitLengthType
7     {
8         /// <summary>
9         /// TestName
10        /// </summary>
11        TestName = 1,
12        //2
13        //4
14    }
15
16    /// <summary>

```

```

17      /// Machine Type
18      /// </summary>
19      public SortedDictionary<ECTIStringLimitLengthType, uint> StringLimitDatas = new
SortedDictionary<ECTIStringLimitLengthType, uint>();
20  }

```

abstract void OnUnknownCommandFeedBack(ArbinCommandUnknownCommandFeed cmd)

- ✓ The callback function for the server's response to unsupported interfaces.

Input:

ArbinCommandUnknownCommandFeed cmd;

```

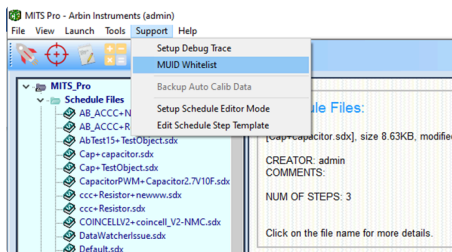
1  public class ArbinCommandUnknownCommandFeed : ArbinCommand
2  {
3      /// <summary>
4      /// CMD
5      /// </summary>
6      public uint UnknownCMD;
7      /// <summary>
8      /// Cmd Extend
9      /// </summary>
10     public uint UnknownCMDExtend;
11 }

```

MUID Verification

MUID verification function basically checks the MUID of the schedule file while assigning schedule or starting/resuming a test. MUID of a schedule file is generated by the hard disk serial number of the local computer on which its created. If a schedule file is uploaded from CTI, a string "From Server" will be added to MUID.

In MitsPro, a MUID whitelist can be added.



Click on MUID Whitelist. A notepad file will open up. Add the hard disk serial numbers, that you want to be recognized by the server. Then save the file.

While assigning/starting/resuming schedule/test through CTI, it will check the MUID of schedule as follows:

```

1  If(MUID == Local hard disk serial number)
2  {
3      pass from checking
4  }
5  Else
6  {
7      If( (MUID == From Server) || (MUID in the whitelist))
8      {
9          pass from checking
10     }

```

```
11 Else
12 {
13     Failed to pass the check, prompt (CTI return reason)
14 }
```