Arbin CTI Protocol Commands

Overall Command Data Format

The Overall command data format consists of **Header + Command Code + Command Args + Checksum.**

The header data will always contain a first byte of 0x11, then 7 bytes of 0xDD, and then the length of the overall command. The command code determines what function to request from the CTI. The command arguments will contain the arguments needed for the specific command code such as specifying which channel to get information from. The checksum is at the end of the command, and is the sum of all bytes of the header + command + command args.

When the CTI server receives a command, it will return a feedback for each command received. The feedback will contain information related to if the command was successfully executed or not. A 10 second timeout is recommended if the client does not receive the feedback associated with the command previously sent.

HEADER DATA

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_HEADER
{
        public ulong Token;      //always contains 0x11dddddddddddddd
        public uint dwLen;       //length of Command + Command Args + Checksum
}
```

`dwLen = Command + Command args + Checksum` for requests

`dwLen = Header + Command + Command args + Checksum` for feedback

COMMAND CODE

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_CMD
{
        public uint dwCmd;
        public uint dwCmd_Extend;
}
```

dwCmd (referred to as the command code) will vary based on the command request. dwCmd_Extend always equals 0x00000000.

COMMAND ARGUMENTS

Argument data will vary based on the command code.

CHECKSUM

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_TAIL_CHECK_CODE
{
        public ushort CheckSum;  //Checksum = add all bytes of header + command + arguments
}
```

COMMANDS

THIRD_PARTY_USER_LOGIN_CMD

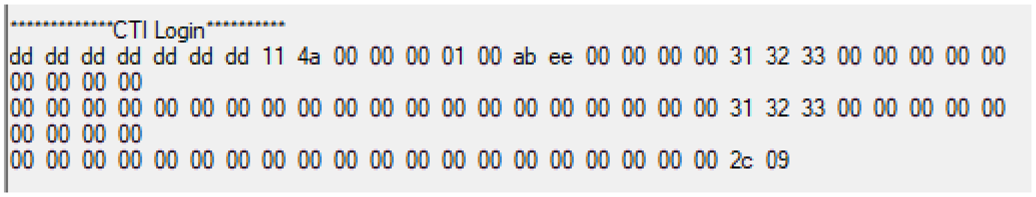
The user must login to the CTI server first before issuing commands to the CTI server.

Command Code = 0XEEAB 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct ARBINVIEWER_USER
{
        public fixed Byte User[32];      //32 characters max
        public fixed Byte Password[32];  //32 characters max
};
*Note: The remaining bytes of user and password need to be filled with 0x00 if the length is less than 32
characters. The same applies to other command fields that require string inputs as well.

[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_USER_LOGIN_CMD
{
        public ARBINVIEWER_USER UserInfo;
}
```



```
*************CTI Login**********
dd dd dd dd dd dd dd dd 11 4a 00 00 00 01 00 ab ee 00 00 00 00 31 32 33 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 32 33 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2c 09
```

Fig1 – example data packet shown for THIRD_PARTY_USER_LOGIN_CMD with User = '123', Password = '123'

Remarks:

None.

THIRD_PARTY_USER_LOGIN_CMD_FEEDBACK

Feedback sent from CTI server after the THIRD_PARTY_LOGIN_CMD is issued. It contains general information about the cycler such as serial number, nickname, location, etc.

Command Code = 0XEEBA 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_USER_LOGIN_FEEDBACK
{
        public uint Result;                       //1 = successful login,2 = failure to login
        public fixed Byte IP_Address[4];
        public fixed Byte SN[16];
        public fixed Byte Note[256];
        public fixed char NickName[1024];
        public fixed char Location[1024];
        public fixed char EmergencyContactNameAndPhoneNumber[1024];
        public fixed char OtherComments[1024];
        public fixed char Email[64];
        public fixed char Call[16];
        public uint ITAC;
        public uint Version;
        public uint IsAllowToControl;
        public uint dwChannelLength;
        public uint dwUserType;
        public uint dwPicLength;
        public fixed char Picture[]
}
```

Remarks:

BYTE Picture[] is variable in length depending on dwPicLength.

THIRD_PARTY_CONNECT

Command used to set/unset the kickout flag. This command is also useful for testing the connection to the CTI server.

Command Code = **0XEEAB 0002**

Command Args

```
[StructLayout( LayoutKind.Sequential, Pack = 1 )]
public unsafe struct THIRD_PARTY_CONNECT
{
        public uint dwSetKickOut;
        public fixed Byte btReserved1[32];
}
```

Remarks:

dwSetKickOut refers to if the login session can be kicked off by another user. 0 for false, 1 for true.

THIRD_PARTY_CONNECT_FEEDBACK

Reports the status of the THIRD_PARTY_CONNECT command.

Command Code = **0XEEBA 0002**

Command Args

```
[StructLayout( LayoutKind.Sequential, Pack = 1 )]
public unsafe struct THIRD_PARTY_CONNECT_FEEDBACK
{
        public uint dwSetKickOut;
        public uint dwConnectResult;     //0: Accepted connection, 1: Rejected connection, it is full
        public fixed Byte btReserved1[32];
}
```

Remarks:

None

THIRD_PARTY_GET_CHANNELS_INFO

This command is used to request channel information from the CTI server.

Command Code = **0XEEAB 0003**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_GET_CHANNELS_INFO
{
        public short OnlyChannel;
        public short InfoType;
        public uint NeedTypeSet;
        public fixed Byte btReserved1[32];
}
```

Remarks:

To get info from all channels on the cycler, set OnlyChannel = -1. A separate feedback will be returned for each channel on the cycler.

The NeedTypeSet parameter indicates additional types of data if they are present on the cycler.

0x000	No auxiliary data required
0x100	CANBMS data required
0x200	SMB data required
0x400	Auxiliary data required

THIRD_PARTY_GET_CHANNELS_INFO_FEEDBACK

Feedback returned from the CTI for the THIRD_PARTY_GET_CHANNELS_INFO command. The data returned is similar to the data found in the Monitor and Control window.

Command Code = **0XEEBA 0003**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_GET_CHANNELS_INFO
{
        public uint ChannelNum; //number of channels
        THIRD_PARTY_CHANNEL Channels[]   //actual length of array determined by ChannelNum
}
```

THIRD_PARTY_CHANNEL contents:

UINT ChannelIndex;

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_STATUS_INFOMATION
{
        public short Status;     //refer to Status codes table
        public byte m_bCommFailure;
        public fixed char Schedule[200];
        public fixed char Testname[72];
        public fixed byte ExitCondition[100];
        public fixed byte StepAndCycleformat[64];
        public fixed char Barcode[72];
        public fixed char CANCfgName[200];
        public fixed char SMBCfgName[200];
        public ushort MasterChannel;
        public double TestTime;
        public double StepTime;
        public float Voltage;
        public float Current;
        public float Power;
        public float ChargeCapacity;
        public float DischargeCapacity;
        public float ChargeEnergy;
        public float DishargeEnergy;
        public float InternalResistance;
        public float dvdt;
        public float ACR;
        public float ACI;
        public float ACIPhase;
        public ushort nAuxVoltageCount;
        public ushort nAuxTemperatureCount;
        public ushort nAuxPressureCount;
        public ushort nAuxExternalCount;
        public ushort nAuxFlowCount;
        public ushort nAuxAoCount;
        public ushort nAuxDiCount;
        public ushort nAuxDoCount;
        public ushort nAuxHumidityCount;
        public ushort nAuxSafetyCount;
        public ushort nAuxPhCount;
        public ushort nAuxDensityCount;
        public ushort BMSNum;
        public ushort SMBNum;
        THIRD_PARTY_AUX_VALUE AuxValues[]; //length determined by nAuxVoltageCount+nAuxTemperatuerCount+..
        THIRD_PARTY_BMS_VALUE BMSValues[]; //length determined by BMSNum
        THIRD_PARTY_SMB_VALUE SMBValues[]; //length determined by SMBNum
}

[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_AUX_VALUE
{
        public float Value;
        public float dtValue;
}

[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_BMS_VALUE
{
        public uint Index;
        public double Value;
}

[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SMB_VALUE
{
        public uint Index;
        public uint nType; //type=0 -> SMBData, type=1 -> SMBString
}
```

Remarks:

Auxiliary, CANBMS, and SMB data is returned in the order of batch file mapping.

THIRD_PARTY_STATUS_INFORMATION Status codes table

0x00	Idle
0x01	Transition
0x02	Charge
0x03	Discharge
0x04	Rest
0x05	Wait
0x06	External Charge
0x07	Calibration
0x08	Unsafe
0x09	Pulse
0x0A	Internal Resistance
0x0B	AC Impedance
0x0C	ACI Cell
0x0D	Test Settings
0x0E	Error

0x0F	Finished
0x10	Volt Meter
0x11	Waiting for ACS
0x12	Pause
0x13	Empty
0x14	Idle from MCU
0x15	Start
0x16	Running
0x17	Step Transfer
0x18	Resume
0x19	Go Pause
0x1A	Go Stop
0x1B	Go Next Step
0x1C	Online Update
0x1D	DAQ Memory Unsafe
0x1E	ACR

THIRD_PARTY_CONTINUE_SCHEDULE

Command used to request the CTI to continue a channel on the cycler. This command is similar to the "Continue" button in the Monitor and Control window.

Command Code = **0XBB32 0006**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_CONTINUE
{
        public uint ChannelNum;
}
```

Remarks:

None

THIRD_PARTY_CONTINUE_SCHEDULE_FEEDBACK

Feedback returned from the CTI after the THIRD_PARTY_CONTINUE_SCHEDULE command is sent by the client that indicates the continue status of the channel requested.

Command Code = **0XBB23 0006**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct
THIRD_PARTY_CONTINUE_FEED_BACK
{
        public int dwIvChannelGlobalIndex;
        public byte btResult;
        public fixed byte Reserved1[101];
}
```

Remarks:

dwIvChannelGlobalIndex = -1 when requested channel successfully starts.

dwIvChannelGlobalIndex = the requested channel if requested channel fails to start.

START_SCHEDULE_FEEDBACK btResult error codes table

CTI_CONTINUE_INDEX	0X10	Invalid channel index
CTI_CONTINUE_ERROR	0X11	There is a user controlling the monitor window (Start/Resume channel window is open)
CTI_CONTINUE_CHANNEL_RUNNING	0X12	Requested channel is running
CTI_CONTINU _CHANNEL_NOT_CONNECT	0X13	Channel not connected to DAQ
CTI_CONTINUE_CHANNEL_CALIBRATING	0X14	Channel Calibrating
CTI_CONTINUE_NOT_PAUSE_NORMAL	0X15	Not Pause Normal
CTI_CONTINUE_CHANNEL_UNSAFE	0X16	Channel is unsafe

THIRD_PARTY_START_SCHEDULE

Command used to request the CTI to start a channel on the cycler. This command is similar to the “Start test” button in the Monitor and Control window.

Command Code = **0XBB32 0004**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_START
{
        public fixed char TestName[72];
        public uint ChannelNum;
}
```

Remarks:

None

THIRD_PARTY_START_SCHEDULE_FEEDBACK

Feedback returned from the CTI after the THIRD_PARTY_START_SCHEDULE command is sent by the client that indicates the start status of the channel requested.

Command Code = **0XBB23 0004**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_START_FEED_BACK
```

```
{
        public int dwIvChannelGlobalIndex;
        public byte btResult;
        public fixed byte Reserved1[101];
}
```

Remarks:

dwIvChannelGlobalIndex = -1 when requested channel successfully starts.

dwIvChannelGlobalIndex = the requested channel if requested channel fails to start.

START_SCHEDULE_FEEDBACK btResult error codes table

CTI_START_INDEX	0X10	Invalid channel index
CTI_START_ERROR	0X11	There is a user controlling the monitor window (Start/Resume channel window is open)
CTI_START_CHANNEL_RUNNING	0X12	Requested channel is running or unsafe
CTI_START_CHANNEL_NOT_CONNECT	0X13	Channel not connected to DAQ
CTI_START_SCHEDULE_VALID	0X14	Schedule not compatible with current system configuration
CTI_START_NO_SCHEDULE_ASSIGNED	0X15	No schedule assigned to channel
CTI_START_SCHEDULE_VERSION	0X16	Schedule version does not match current version of MITS
CTI_START_POWER_PROTECTED	0X17	Not used
CTI_START_RESULTS_FILE_SIZE_LIMIT	0X18	Not used
CTI_START_STEP_NUMBER	0X19	Invalid step number
CTI_START_NO_CAN_CONFIGURATION_ASSIGNED	0X1A	Not used
CTI_START_AUX_CHANNEL_MAP	0X1B	Invalid auxiliary count in schedule
CTI_START_BUILD_AUX_COUNT	0X1C	Invalid build in auxiliary count
CTI_START_POWER_CLAMP_CHECK	0X1D	Not used
CTI_START_AI	0X1E	Check Aux Test Setting tab
CTI_START_SAFOR_GROUPCHAN	0X1F	No selected channels
CTI_START_BT6000RUNNINGGROUP	0X20	
CTI_START_CHANNEL_DOWNLOADING_SCHEDULE	0X21	DAQ still downloading schedule
CTI_START_DATABASE_QUERY_TEST_NAME_ERROR	0X22	Error querying database (database connection closed most likely)
CTI_START_TEXTNAME_EXISTS	0X23	Testname cannot be empty or schedule does not match last used

		schedule in the case of resuming
CTI_START_GO_STEP	0X24	Invalid step number
CTI_START_INVALID_PARALLEL	0X25	Invalid parallel channel number
CTI_START_SAFETY	0X26	Schedule safety precheck failed
CTI_START_SCHEDULE_NAME_DIFFERENT	0X27	Not used
CTI_START_BATTERYSIMULATION_NOT_PARALLEL	0X28	Battery simulation error

THIRD_PARTY_RESUME_SCHEDULE

Command used to resume a schedule. This command is similar to the "resume test" button in the Monitor and Control window.

Command Code = 0XBB31 0002

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_RESUME
{
        public int dwIvChannelGlobalIndex;
        public byte ResumeAll;
        public fixed byte Reserved1[101];
}
```

Remarks:

See THIRD_PARTY_RESUME_SCHEDULE_FEEDBACK.

THIRD_PARTY_RESUME_SCHEDULE_FEEDBACK

Feedback returned from the CTI after the THIRD_PARTY_RESUME_SCHEDULE command is sent by the client that indicates the resume status of the requested channel.

Command Code = 0XBB13 0002

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_RESUME_FEED_BACK
{
        public int dwIvChannelGlobalIndex;
        public byte btResult;
        public fixed byte Reserved1[101];
}
```

Remarks:

If ResumeAll is used in THIRD_PARTY_RESUME_SCHEDULE, an individual feedback is returned for each channel.

RESUME_SCHEDULE_FEEDBACK btResult error codes table

CTI_START_INDEX	0X10	Invalid channel index

CTI_START_ERROR	0X11	There is a user controlling the monitor window (Start/Resume channel window is open)
CTI_START_CHANNEL_RUNNING	0X12	Requested channel is running or unsafe
CTI_START_CHANNEL_NOT_CONNECT	0X13	Channel not connected to DAQ
CTI_START_SCHEDULE_VALID	0X14	Schedule not compatible with current system configuration
CTI_START_NO_SCHEDULE_ASSIGNED	0X15	No schedule assigned to channel
CTI_START_SCHEDULE_VERSION	0X16	Schedule version does not match current version of MITS
CTI_START_POWER_PROTECTED	0X17	Not used
CTI_START_RESULTS_FILE_SIZE_LIMIT	0X18	Not used
CTI_START_STEP_NUMBER	0X19	Invalid step number
CTI_START_NO_CAN_CONFIGURATION_ASSIGNED	0X1A	Not used
CTI_START_AUX_CHANNEL_MAP	0X1B	Invalid auxiliary count in schedule
CTI_START_BUILD_AUX_COUNT	0X1C	Invalid build in auxiliary count
CTI_START_POWER_CLAMP_CHECK	0X1D	Not used
CTI_START_AI	0X1E	Check Aux Test Setting tab
CTI_START_SAFOR_GROUPCHAN	0X1F	No selected channels
CTI_START_BT6000RUNNINGGROUP	0X20	
CTI_START_CHANNEL_DOWNLOADING_SCHEDULE	0X21	DAQ still downloading schedule
CTI_START_DATABASE_QUERY_TEST_NAME_ERROR	0X22	Error querying database (database connection closed most likely)
CTI_START_TEXTNAME_EXISTS	0X23	Testname cannot be empty or schedule does not match last used schedule in the case of resuming
CTI_START_LOAD_RESUME	0X24	Not used
CTI_START_MAX_MULTIPLE_RESULT	0X25	Not used
CTI_START_SAFETY	0X26	Schedule safety precheck failed
CTI_START_BATTERYSIMULATION_NOT_PARALLEL	0X27	Battery simulation error

THIRD_PARTY_STOP_SCHEDULE

Command used to stop a running test. This command is similar to the “Stop test” button in the Monitor and Control window.

Command Code = **0XBB31 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_STOP
{
        public uint m_dwlvChannelGlobalIndex;
        public Byte m_btStopAll;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

None

THIRD_PARTY_STOP_SCHEDULE_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_STOP_SCHEDULE command is sent by the client that indicates the stop status of the requested channel.

Command Code = **0XBB13 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SCHEDULE_STOP_FEED_BACK
{
        public int m_dwlvChannelGlobalIndex;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

If StopAll is used in THIRD_PARTY_STOP_SCHEDULE, a separate feedback is returned for each channel.

STOP_SCHEDULE_FEEDBACK m_btResult error codes table

CTI_STOP_INDEX	0x10	Channel index does not exist
CTI_STOP_ERROR	0x11	Someone else is controlling monitor window at the moment
CTI_STOP_NOT_RUNNING	0x12	Not used
CTI_STOP_CHANNEL_NOT_CONNECT	0x13	Not used

THIRD_PARTY_ASSIGN_SCHEDULE

Command used to assign a schedule to a channel. This command is similar to assigning schedules in the Monitor and Control window.

Command Code = **0XBB21 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_ASSIGN_SDU
{
        public int dwlvChannelGlobalIndex;
        public byte AssignSduAll;
        public fixed char ScheduleName[200];
        public float flCapacity;
        public fixed char ItemId[72];
        public float fMV_UD1;
        public float fMV_UD2;
        public float fMV_UD3;
        public float fMV_UD4;
        public float fMV_UD5;
        public float fMV_UD6;
        public float fMV_UD7;
        public float fMV_UD8;
        public float fMV_UD9;
        public float fMV_UD10;
        public float fMV_UD11;
        public float fMV_UD12;
        public float fMV_UD13;
        public float fMV_UD14;
        public float fMV_UD15;
        public float fMV_UD16;
        public fixed byte Reserved1[32];
}
```

Remarks:

The parameters flCapacity to fMV_UD16 are used in MITS7 only. Using these parameters in MITS8 does not have any effect.

THIRD_PARTY_ASSIGN_SCHEDULE_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_ASSIGN_SCHEDULE command is sent by the client. It indicates the schedule assignment status of the channel requested.

Command Code = **0XBB12 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_ASSIGN_SDU_FEED_BACK
{
        public int dwlvChannelGlobalIndex;
        public byte btResult;
        public fixed byte Reserved1[101];
}
```

Remarks:

A separate feedback is sent for each channel if “AssignSduAll” = 1 in the THIRD_PARTY_ASSIGN_SCHEDULE command.

ASSIGN_SCHEDULE_FEEDBACK error codes table

CTI_ASSIGN_INDEX	0x10	Channel does not exist

CTI_ASSIGN_ERROR	0x11	Monitor window in use at the moment
CTI_ASSIGN_SCHEDULE_NAME_EMPTY_ERROR	0x12	Schedule name cannot be empty
CTI_ASSIGN_SCHEDULE_NOT_FIND_ERROR	0x13	Schedule name not found
CTI_ASSIGN_CHANNEL_RUNNING_ERROR	0x14	Channel is running
CTI_ASSIGN_CHANNEL_DOWNLOAD_ERROR	0x15	Channel is downloading another schedule currently
CTI_ASSIGN_BATCH_FILE_OPENED	0x16	Cannot assign schedule when batch file is open
CTI_ASSIGN_SDU_CANNOT_ASSIGN_SCHEDULE	0x17	Assign failed
CTI_ASSIGN_SDU_SAVE_FAILED	0x18	Not used

THIRD_PARTY_SET_MV_VALUE

Command used to set the user defined meta-variables found in the “Log and Others” tab of the schedule editor.

Command Code = 0XBB15 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SET_MV_VALUE
{
        public uint m_dwIvChannelGlobalIndex;
        public int MV_Type;
        public int MV_MetaCode;
        public fixed Byte m_btReserved1[16];
        public int MV_ValueType;
        public float MV_Data;
        public fixed Byte m_btReserved2[16];
}
```

Remarks:

The channel must be running for this command to take effect. CTI only allows MV_Type = 1, and MV_ValueType = 1.

The MV_MetaCode values for MV_UD1-16 located in the schedule editor are shown in the table below:

MetaCode_MV_UD1	52 (decimal)
MetaCode_MV_UD2	53
MetaCode_MV_UD3	54
MetaCode_MV_UD4	55
MetaCode_MV_UD5	105
MetaCode_MV_UD6	106
MetaCode_MV_UD7	107
MetaCode_MV_UD8	108
MetaCode_MV_UD9	109

MetaCode_MV_UD10	110
MetaCode_MV_UD11	111
MetaCode_MV_UD12	112
MetaCode_MV_UD13	113
MetaCode_MV_UD14	114
MetaCode_MV_UD15	115
MetaCode_MV_UD16	116

THIRD_PARTY_SET_MV_VALUE_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_SET_MV_VALUE command is sent by the client. It indicates the status of setting the metavariables for the requested channel.

Command Code = **0XBB51 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SET_MV_Value_FEED_BACK
{
        public int m_dwlvChannelGlobalIndex;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

SET_MV_VALUE b_btResult values.

CTI_SET_MV_ERROR	0x10	Set MV failed
CTI_SET_MV_METACODE_NOTEXIST	0x11	MV meta code does not exist
CTI_EST_MV_CHANNEL_NOT_STARTED	0x12	Channel is not running
CTI_SET_MV_METACODE_NOTEXIST_Pro7	0x13	This meta code does not exist in Mits Pro7

THIRD_PARTY_UPDATE_MV_ADVANCED

Command used to set the user defined meta-variables found in the “Log and Others” tab of the schedule editor.

Command Code = **0XBB15 0002**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public  unsafe struct  METAVARIABLE_DATA_CH_CODE
 {
          public ushort  m_ChannelIndexInGlobal;
          public ushort  m_MV_MetaCode;
          public float  fMV_Value;
 }

[StructLayout(LayoutKind.Sequential,  Pack = 1)]
public unsafe struct  THIRD_PARTY_UPDATE_META_VARIABLE_ADVANCED_CODE
 {
          public ushort  m_nMV_Total;
          public fixed  Byte m_btReserved1[18];
          public METAVARIABLE_DATA_CH_CODE  m_MV_Data[MAX_METAVARIABLE_SINGLE_PACK];
 }
```

Remarks:

The MV_MetaCode values for MV_UD1-16 located in the schedule editor are shown in the table below:

MetaCode_MV_UD1	52 (decimal)
MetaCode_MV_UD2	53
MetaCode_MV_UD3	54
MetaCode_MV_UD4	55
MetaCode_MV_UD5	105
MetaCode_MV_UD6	106
MetaCode_MV_UD7	107
MetaCode_MV_UD8	108
MetaCode_MV_UD9	109

MetaCode_MV_UD10	110
MetaCode_MV_UD11	111
MetaCode_MV_UD12	112
MetaCode_MV_UD13	113
MetaCode_MV_UD14	114
MetaCode_MV_UD15	115
MetaCode_MV_UD16	116

THIRD_PARTY_UPDATE_MV_ADVANCED_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_SET_MV_VALUE command is sent by the client. It indicates the status of setting the metavariables for the requested channel.

Command Code = **0XBB51 0002**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SET_MV_Value_FEED_BACK
{
        public int m_dwIvChannelGlobalIndex;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

SET_MV_VALUE b_btResult values.

CTI_SET_MV_ERROR	0x10	Set MV failed
CTI_SET_MV_METACODE_NOTEXIST	0x11	MV meta code does not exist
CTI_EST_MV_CHANNEL_NOT_STARTED	0x12	Channel is not running
CTI_SET_MV_METACODE_NOTEXIST_Pro7	0x13	This meta code does not exist in Mits Pro7
CTI_SET_MV_METACODE_UPDATE_TOO_FREQUENTLY_200MS	0x14	Update too frequently (updated every 200 milliseconds)

THIRD_PARTY_JUMP_CHANNEL

Command used to request a running channel to jump to a particular step in its schedule. This command is similar to the “Jump Step” button in the Monitor and Control window.

Command Code = **0XBB32 0005**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_JUMP_CHANNEL
{
        public int m_stepNum;
        public int m_channelIndex;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

m_stepNum is 0-indexed. If a step number is given and it does not exist in the schedule, the test will stop automatically.

THIRD_PARTY_JUMP_CHANNEL_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_JUMP_CHANNEL command is sent by the client. It indicates whether a channel has successfully jumped to another step or not in its schedule.

Command Code = **0XBB23 0005**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_JUMP_CHANNEL_FEED_BACK
{
        public int dwIvChannelGlobalIndex;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

Sending a step number greater than the number of steps in the schedule will stop the test. Step numbers are 0-indexed.

JUMP_CHANNEL m_btResult error codes table:

CTI_JUMP_INDEX	0x10	Not used
CTI_JUMP_ERROR	0x11	Someone else is using the monitor window at the moment
CTI_JUMP_CHANNEL_RUNNING	0x12	Channel not running
CTI_JUMP_CHANNEL_NOT_CONNECT	0x13	Channel not connected to DAQ
CTI_JUMP_SCHEDULE_VALID	0x14	Invalid schedule
CTI_JUMP_NO_SCHEDULE_ASSIGNED	0x15	No schedule assigned
CTI_JUMP_SCHEDULE_VERSION	0x16	Invalid schedule version
CTI_JUMP_POWER_PROTECTED	0x17	Not used
CTI_JUMP_RESULTS_FILE_SIZE_LIMIT	0x18	Not used
CTI_JUMP_STEP_NUMBER	0x19	Schedule cannot contain over 200 steps
CTI_JUMP_NO_CAN_CONFIGURATION_ASSIGNED	0x1A	Not used
CTI_JUMP_AUX_CHANNEL_MAP	0x1B	Not used
CTI_JUMP_BUILD_AUX_COUNT	0x1C	Not used
CTI_JUMP_POWER_CLAMP_CHECK	0x1D	Not used
CTI_JUMP_AI	0x1E	Not used
CTI_JUMP_SAFOR_GROUPCHAN	0x1F	Not used
CTI_JUMP_BT6000RUNNINGGROUP	0x20	Not used
CTI_JUMP_CHANNEL_DOWNLOADING_SCHEDULE	0x21	DAQ still downloading schedule
CTI_JUMP_DATABASE_QUERY_TEST_NAME_ERROR	0x22	Not used
CTI_JUMP_TESTNAME_EXISTS	0x23	Not used
CTI_JUMP_GO_STEP	0x24	Schedule contains invalid step limit setting
CTI_JUMP_INVALID_PARALLEL	0x25	Invalid parallel setting
CTI_JUMP_SAFETY	0x26	Schedule safety check not safe

CTI_JUMP_SCHEDULE_NAME_DIFFERENT	0x27	Not used
CTI_JUMP_BATTERYSIMULATION_NOT_PARALLEL	0x28	Battery simulation not parllel

THIRD_PARTY_BROWSE_DIRECTORY

This command allows you to browse the valid folders and files inside the MITS_PRO directory.

Command Code = 0XCC13 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_BROWSE_DIRECTORY
{
        public fixed char DirectoryPath[1024];
}
```

Remarks:

Valid directories include the Simulation, Work, Support, SMB Config, and CANBMS Config paths. The Directorypath is limited to 1024 characters.

THIRD_PARTY_BROWSE_DIRECTORY_FEED

Feedback returned by the CTI server after the THIRD_PARTY_BROWSE_DIRECTORY command is sent by the client. It contains file directory information, and error code if any.

Command Code = 0XCC31 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_BROWSE_DIRECTORY_FEED
{
        public uint Result;
        public uint nDirFileCount;      //total number of files and directories
        public DirFileInfo DirFileList[]
}
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct DirFileInfo
{
        public uint Type;                //0: directory, 1: file
        public fixed char DirFileName[64];
        public int dwSize;               //file size
        public fixed char wcModified[32];  //last time modified
}
```

Remarks:

BROWSE_DIRECTORY Result values:

CTI_BROWSE_DIRECTORY_SUCCESS	0x1	Valid directory, return success
CTI_BROWSE_SCHEDULE_SUCCESS	0x2	Schedule directory, return success
CTI_BROWSE_DIRECTORY_FAILED	0x3	Invalid directory, return failed

THIRD_PARTY_DOWNLOAD_FILE

This command allows the client to download a file from the MITS_PRO directory on the CTI server side.

Command Code = **0XCC13 0002**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_DOWNLOAD_CODE
{
        public fixed char FilePath[THIRD_PARTY_PATH_MAX_LENGHT_CONTAIN_ZERO];
        public double DownloadTime;       //timestamp
}
```

Remarks:

None

THIRD_PARTY_DOWNLOAD_FILE_FEEDBACK

This command contains the file data and extra information needed to receive the file client side after the THIRD_PARTY_DOWNLOAD_FILE command is sent by the client.

Command Code = **0XCC31 0002**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_DOWNLOAD_FEED_CODE
{
        public uint Result;
        public double DownloadTime;
        public uint uGeneralPackage;      //Number of total packages to be sent
        public uint uPackageIndex;        //Current package index
        public fixed byte m_MD5[16];      //MD5
        public ulong ulFileLength;        //File data comes after uFileLength
}
```

Remarks:

File packages are broken into chunks of 512 KB.

Compute the MD5 of the file chunk received with the m_MD5 field to check for data consistency.

DOWNLOAD_FILE_FEEDBACK Result error codes table

CTI_DOWNLOAD_SUCCESS	1	File download success
CTI_DOWNLOAD_FAILED	2	Error in file download
CTI_DOWNLOAD_MD5_ERR	3	MD5 hash of file data does not match

CTI_DOWNLOAD_MAX_LENGTH_ERR	4	File size exceeded

THIRD_PARTY_UPLOAD_FILE

This command is used to upload a file from client side to the CTI server side.

Command Code = **0XCC13 0003**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_UPLOAD_CODE
{
        public fixed char FilePath[THIRD_PARTY_PATH_MAX_LENGHT_CONTAIN_ZERO];
        public ulong ulFileLength;
        public uint uGeneralPackage;
        public uint uPackageIndex;
        public double UploadTime;
        public fixed byte m_MD5[16];
}
```

Remarks:

File chunks can be broken up to different sizes with the chunk size specified in ulFileLength.

THIRD_PARTY_UPLOAD_FILE_FEEDBACK

Command Code = **0XCC31 0003**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_UPLOAD_FEED_CODE
{
        public double UploadTime;
        public uint uGeneralPackage;
        public uint uPackageIndex;
        public uint Result;
}
```

Remarks:

None

UPLOAD_FILE_FEEDBACK Result error codes table

CTI_UPLOAD_SUCCESS	1	Upload to CTI success
CTI_UPLOAD_FAILED	2	Upload to CTI failed
CTI_UPLOAD_MD5_ERR	3	MD5 mismatch

THIRD_PARTY_NEW_OR_DELETE_FOLDER

Command used for creating or folder or deleting a folder/file in the MITS_PRO directory on the CTI server.

Command Code = **0XCC13 0004**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_NEW_OR_DELETE_CODE
{
        public uint Type;        //0: delete, 1: create
        public fixed char FilePath[1024];
}
```

Remarks:

This command only allows to create new folders, but allows to delete files/folders. Use the THIRD_PARTY_UPLOAD_FILE command to create and upload a new file.

Only allows deleting files ending in .sdx, .sdu, .can, .smb, and .txt file extension. FilePath is limited to 1024 characters.

THIRD_PARTY_NEW_OR_DELETE_FOLDER_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_NEW_OR_DELETE_FOLDER command is sent by the client. It contains the error code for the command request.

Command Code = **0XCC31 0004**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_NEW_OR_DELETE_FEED_CODE
{
        public uint Result;
}
```

Remarks:

NEW_OR_DELETE_FOLDER_FEEDBACK Result error codes table

CTI_NEW_SUCCESS	0x1	New file success
CTI_DELETE_SUCCESS	0x2	Delete file success
CTI_NEW_FAILED	0x3	New file failed
CTI_NEW_FAILED_ADD_FOLDER	0x4	Cannot add folders in this directory
CTI_DELETE_FAILED	0x5	Delete file failed
CTI_DELETE_FAILED_EXTENSION	0x6	Cannot delete files of this type
CTI_DELETE_FAILED_TEXT_RUNNING	0x7	File is in use
CTI_DELETE_FAILED_EXIST	0x8	File does not exist

THIRD_PARTY_NEW_DIRECTORY

Specific command used to create new folders only.

Command Code = **0XCC13 0005**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_NEW_FOLDER_CODE
{
        public fixed char FilePath[1024];
}
```

Remarks:

FilePath is limited to 1024 characters.

THIRD_PARTY_NEW_DIRECTORY_FEEDBACK

Feedback returned by the CTI server indicating the success status of the THIRD_PARTY_NEW_DIRECTORY command.

Command Code = **0XCC31 0005**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_NEW_FOLDER_CODE
{
        public int Result;
}
```

Remarks:

See THIRD_PARTY_NEW_OR_DELETE_FOLDER remarks for the error codes returned in Result.

THIRD_PARTY_GET_SERIAL

Command Code = **0XBB34 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_GET_SERIAL
{
        public uint m_dwGetSerialNum;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

None

THIRD_PARTY_GET_SERIAL_FEEDBACK

Command Code = **0XBB43 0001**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_GET_SERIAL_FEED_BACK
{
        public uint m_dwGetSerialNum;
        public Byte m_btResult;
        public fixed Byte m_btReserved1[101];
}
```

Remarks:

None

THIRD_PARTY_DELETE_DIRECTORY

Specific command used to delete a file or folder from the MITS_PRO directory on the CTI server.

Command Code = **0XCC13 0006**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_NEW_FOLDER_CODE
{
        public fixed char FilePath[1024];
}
```

Remarks:

The FilePath is limited to 1024 characters.

THIRD_PARTY_DELETE_DIRECTORY_FEEDBACK

Feedback returned by the CTI server after the THIRD_PARTY_DELETE_DIRECTORY command is sent by the client. It indicates the success status of the command request.

Command Code = **0XCC31 0006**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_DELETE_FOLDER_FEED_CODE
{
        public uint Result;
}
```

Remarks:

See THIRD_PARTY_NEW_OR_DELETE_FOLDER remarks for the error codes returned in Result.

THIRD_PARTY_SEND_MSG_TO_CTI_CODE

This command is used to send a message to the CTI server. A message popup window will show on the CTI server when it receives the message.

Command Code = **0XCD14 0002**

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SEND_MSG_TO_CTI_CODE
{
        public fixed char Msg[1024];
}
```

Remarks:

Message is limited to 1024 characters.

THIRD_PARTY_SEND_MSG_TO_CTI_FEED_CODE

Feedback returned by the CTI server indicating if the THIRD_PARTY_SEND_MSG_TO_CTI_CODE command was successfully received from the client.

Command Code = 0XCD41 0002

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_SEND_MSG_TO_CTI_FEED_CODE
{

        public uint Result;
};
```

Remarks:

Result = 0x1; successfully sent and received by CTI

THIRD_PARTY_AUTOCALI_START_CODE

Command used to start auto calibration on the cycler connected to MITS running the CTI server.

Command Code = 0XCD14 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_AUTOCALI_START_CODE
{
        public fixed Byte Reserved[102];
}
```

Remarks:

This command starts auto-calibration. The Calibration window must be opened when this command is used.

THIRD_PARTY_AUTOCALI_START_FEED_CODE

Feedback returned from the CTI server indicating whether auto calibration has started successfully or not.

Command Code = 0XCD41 0001

Command Args

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public unsafe struct THIRD_PARTY_AUTOCALI_START_CODE
{
        public uint Result; //0: start failed, 1: start success
}
```

Remarks:

None