



## ArbinCTI Manual 1.2.0

### [Introduction](#)

### [How to Enable CTI in MitsPro](#)

### [How to use CTI library for software development](#)

### [ArbinCTI APIs](#)

#### [Connection](#)

`int ConnectAsync (string strIPv4, int Port, int TimeOut, out int ErrorCode)`

`bool IsConnected ()`

`void ShutDown ()`

#### [Initialization and Exiting](#)

`void Start ()`

`void ListenSocketRecv (IArbinSocket socket)`

`void Exit ()`

`string GetCTIVersion()`

#### [Login Commands](#)

`bool PostLogicConnect (IArbinSocket socket, bool bSetKickOut);`

`bool PostUserLogin (IArbinSocket socket, string strUser, string strPassword);`

#### [Test Schedule Commands](#)

`bool PostModifySchedule(IArbinSocket socket, ModifyScheduleArgs args)`

`bool PostAssignSchedule (IArbinSocket socket, string ScheduleName, String Barcode, float Capacity, float MVUD1, float MVUD2, float MVUD3, float MVUD4, [bool AllAssign = true], [int ChannelIndex = -1])`

`bool PostSetMetaVariable (IArbinSocket socket, int nChannel, int mvType, int mvMetaCode, int mvValueType, object mvValue)`

`bool PostUpdateMetaVariableAdvanced (IArbinSocket socket, List<Meta VariableInfo> metaVariableList, out int error)`

`bool PostUpdateMetaVariableAdvancedEx (IArbinSocket socket, List<MetaVariableInfoEx> metaVariableList, out int error)`

`public bool PostTimeSensitiveSetMV(IArbinSocket socket, TimeSensitiveSetMVArgs args)`

`bool PostGetMetaVariables(IArbinSocket socket, List<ArbinCommandGetMetaVariablesFeed.MetaVariableInfo> metaVariableList, out int error)`

`bool PostSetIntervalTimeLogData (IArbinSocket socket, float fIntervalTime)`

`bool PostApplyForUDPCCommunication(IArbinSocket socket, ushort uUDPCClientPort, string strUDPCClientIP, List<CMetavariabDataCodeApply> dataCodeApplies)`

`bool PostUpdateParameters(IArbinSocket socket, List<ushort> chanList, bool isForAllChans, SortedDictionary<ushort, double> parameters)`

`bool PostConvertToAnonymousOrNamedTO(IArbinSocket socket, bool to Anonymous, bool isForAll, List<ushort> channelList)`

`bool PostAssignBarcodeInfo(IArbinSocket socket, ArbinCommandAssignBarcodeInfoFeed.EChannelType channelType, List<ArbinCommandAssignBarcodeInfoFeed.ChannelBarcodeInfo> barcodeInfos)`

`bool PostGetBarcodeInfo(IArbinSocket socket, ArbinCommandGetBarcodeInfoFeed.EChannelType channelType, List<ArbinCommandGetBarcodeInfoFeed.GetChannelBarcodeInfo> barcodeInfos)`

`bool PostGetMachineType(IArbinSocket socket)`

`bool PostGetTrayStatus(IArbinSocket socket, List<ushort> trayIndexList, out int error)`

```
bool PostEngageTray(IArbinSocket socket, List<ArbinCommandEngageTrayFeed.EngageTrayInfo> engageTrayList, out int error)
```

#### Channel Controls

```
bool PostStartChannel (IArbinSocket socket, string TestName, List<ushort> Channels);  
bool PostStartChannelEx (IArbinSocket socket, List<StartResumeEx> resumeEx, string Creators, string Comments)  
bool PostStartChannelAdvanced(IArbinSocket socket, StartChannelAdvancedArgs args)  
bool PostStopChannel (IArbinSocket socket, int nChannel, bool bStopAll)  
bool PostJumpChannel (IArbinSocket socket, int stepNum, int channelIndex)  
bool PostResumeChannel (IArbinSocket socket, bool AllResume, int Channel Index)  
bool PostResumeChannelEx (IArbinSocket socket, List<StartResumeEx> resumeExs)  
bool PostResumeChannelByMetaVariables(IArbinSocket socket, List<StartResumeEx> resumeExs)  
bool PostContinueChannel(IArbinSocket socket, List<ushort> Channels)
```

#### File Operation Commands

```
bool PostBrowseDirectory (IArbinSocket socket, string strPath)  
bool PostDeleteFolder (IArbinSocket socket, string strPath)  
bool PostDownloadFile (IArbinSocket socket, string strPath, double time)  
bool PostNewFolder (IArbinSocket socket, string strPath)  
bool PostNewOrDelete (IArbinSocket socket, string strPath, ArbinCommand NewOrDeleteFeed.NEW_OR_DELETE_TYPE uType)  
bool PostUpLoadFile (IArbinSocket socket, string strPath, byte[] Filedata, double time, uint uGeneralPackage, uint PackageIndex)  
bool PostUpLoadFile(IArbinSocket socket, string strRemoteRelativeFileName , string strLocalFullFileName, bool bOverride, ArbinCommandUpLoadFile Feed.CUpLoadFileResult.AsyncCallback uploadFileResultCallback)  
bool PostCheckFileEx(IArbinSocket socket, string strRemoteRelativeFile Name, byte[] MD5);  
bool PostAssignFile(IArbinSocket socket, string fileName, bool bAssignAll, ArbinCommandAssignFileFeed.EFileKind eFileKind, List<ushort> chanList)
```

#### Messaging Commands

```
bool PostSendMsgToCTI (IArbinSocket socket, string msg)
```

#### Calibration Commands

```
bool PostStartAutomaticCalibration (IArbinSocket socket)
```

#### Request Information Commands

```
bool PostGetMappingAux (IArbinSocket socket, GetMappingAuxArgs args)  
bool PostGetChannelsData (IArbinSocket socket, uint NeedType, short OnlyGet ChannelNumber, ArbinCommandGetChannelFeed.GET_CHANNEL_TYPE GetChannelType)  
bool PostGetChannelsDataSPTT(IArbinSocket socket, short OnlyGetChannelNumber= -1)  
bool PostGetChannelsDataMminimalistMode(IArbinSocket socket, short onlyGetChannelNumber = -1)  
bool PostGetChannelsDataSimpleMode(IArbinSocket socket, short OnlyGet ChannelNumber = -1, ArbinCommandGetChannelDataSimpleModeFeed. GET_CHANNEL_TYPE GetChannelType = ArbinCommandGetChannelDataSimpleModeFeed.GET_CHANNEL_TYPE.ALLCHANNEL)  
bool PostGetResumeData (IArbinSocket socket, List<ushort> channels)  
bool PostGetSerialNumber(IArbinSocket socket)  
bool PostGetStartData (IArbinSocket socket, List<ushort> channels)  
bool PostGetServerSoftwareVersionNumber(IArbinSocket socket);  
bool PostGetStringLimitLength(IArbinSocket socket, uint NeedType = (uint)ArbinCommandGetStringLimitLengthFeed.ECTIStringLimitLengthType.TestName))  
Task<GetUDSSignalValueResponse> SendGetUDSSignalValue(IArbinSocket socket, GetUDSSignalValueRequest request)  
Task<SetRequestUDSSignalValueResponse> SendSetUDSSignalValue(IArbinSocket socket, SetRequestUDSSignalValueRequest request)  
Task<SendUDSMessageResponse> SendSendUDSMessage(IArbinSocket socket, SendUDSMessageRequest request)  
Task<ExecUDSUnlockResponse> SendExecUDSUnlock(IArbinSocket socket, ExecUDSUnlockRequest request)
```

## Introduction

CTI is an abbreviation of Console TCP/IP Interface. It provides an interface for the third-party App to communicate with the Arbin battery testing system. It defines the protocol of information exchanges between Arbin MitsPro software and third-party applications. MitsPro software will serve as a server and a third-party App will be a client.

Authorized users are allowed to send requests to MitsPro. For each legitimate request, MitsPro will execute the request and send the feedback of the request to the client asynchronously. Both the request and feedback use TCP/IP communication protocol.

The commands that the Client send to MitsPro can be classified into two categories:

The first category is those who need MitsPro to take action to control the cyclers, for example, to assign a schedule to channels, start, jump, stop, or resume channels. These commands will trigger MitsPro to send commands to microcontrollers in cyclers and eventually will change the running status of the related channels.

The second category is those requiring MitsPro to provide the status and other information about channels. These commands will not change the running state of channels.

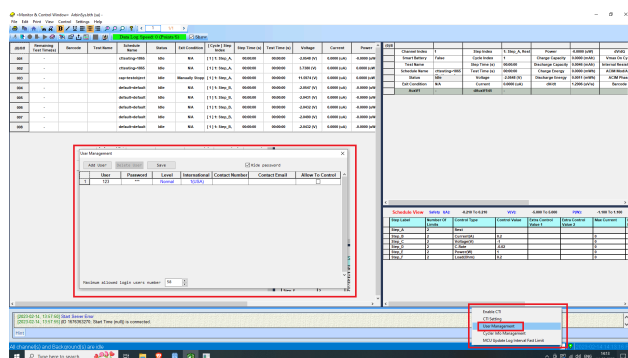
**!** It is recommended to use the C# language for the interface program to ensure that it is consistent with the DLL development language to avoid any complications.

ArbinCTI.DLL is developed on the .NET framework 4.0, and the System.Drawing.Image package is called to send and receive pictures. If third-party software is developed on the .net5 or .net6 platform, you need to install the system.drawing.common package separately.

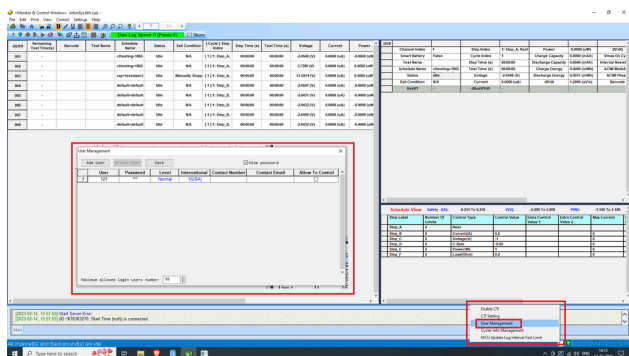
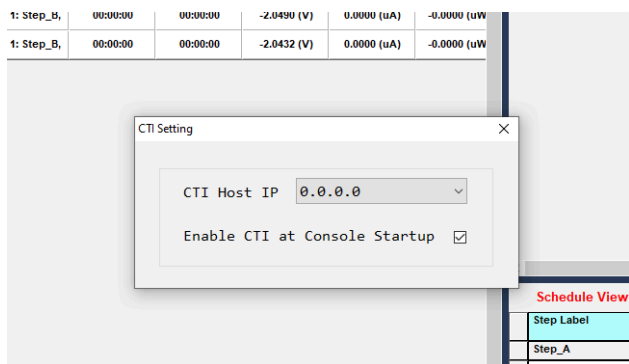
**i** Hereafter, in this document MitsPro will be referred as server and ArbinCTI as client.

## How to Enable CTI in MitsPro

- Open MitsPro Software
- Open Monitor and Control Window
- Go to CTI settings in the Monitor window (If CTI settings are not accessible, disable CTI)

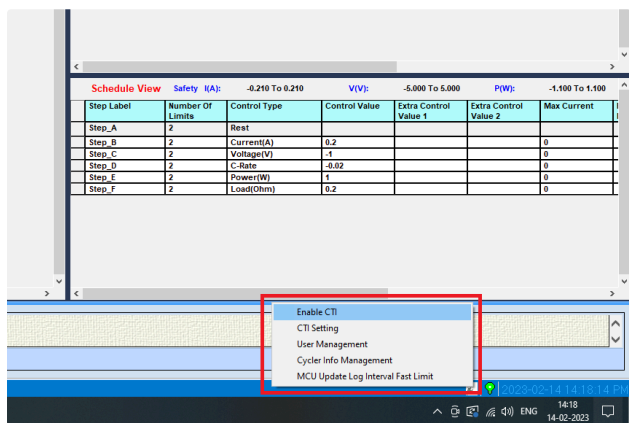


- In CTI settings, Select the server host IP. (Network IP of computer running MitsPro software)
- Open User Management
- Create Users and Save

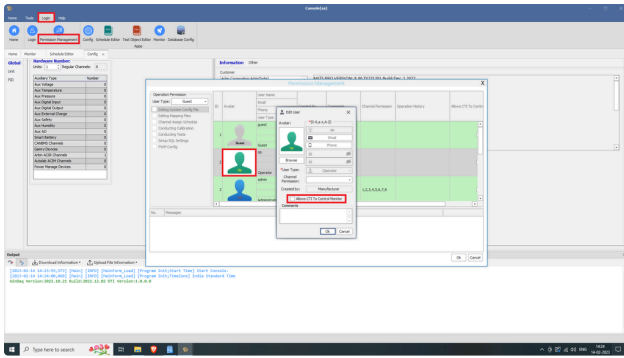


⚠ If “Allow to Control” box is checked for a user then only that user can control the cyclor. Though other users can login but can only view the channel status. Only one user at a time can control the cyclor.

- Finally Enable CTI from CTI settings.



📌 In MitsX CTI is by default enabled. One just need to allow the user to control through CTI. Go to Login => Permission Management. Double click on user to whom you want to give permissions. Check the check box – Allow CTI to control monitor.



## How to use CTI library for software development

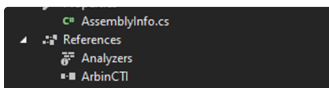
It is recommended to use the C# language for the interface program to ensure that it is consistent with the DLL development language to avoid any complications.

ArbinCTI.DLL is developed on the .NET framework 4.0, and the System.Drawing.Image package is called to send and receive pictures. If third-party software is developed on the .net5 or .net6 platform, you need to install the system.drawing.common package separately.

Import following namespaces of ArbinCTI.dll

```
using ArbinCTI.Core.Control;
using ArbinCTI.Core.Interface;
```

Add reference of ArbinCTI.dll to your project in your IDE.



## ArbinCTI APIs

### Connection

To execute any of the CTI commands(functions), connection must be established with the server.

#### **int ConnectAsync (string strIPV4, int Port, int TimeOut, out int ErrorCode)**

✓ This command establishes a connection with the server.

#### **Prerequisite:**

✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Following objects needs to be defined.

```
ArbinClient myClient = new ArbinClient ();
```

```
string myIP = "CTI host IP"; (Which was set in CTI settings on server)
```

```
int myErrorCode, myTimeOut=0, myPort = 9031/9032;
```

In order to establish a connection, the Monitor and Control window must be open on the server. (In case of MitsX Console and Daq both must be open)

**Usage:**

```
myClient.ConnectAsync (myIP, myPort, myTimeOut, out myErrorCode);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
strIPV4	Host IP that you set in CTI settings	string	Actual IP address
Port	Default server's listening port (Default Value - Cannot be changed)	int	9031 - used for cyclor control 9032 - used for file sharing
TimeOut	Time (seconds) to wait for connection establishment	int	0 - wait Infinite n - n seconds (n is any +ve integer)

**Output:**

Parameter Name	Description	Return type	Valid Values
ErrorCode	Connection status	int	0 - connection successful -4 - connection time out

## bool IsConnected ()

- ✓ To Check whether the ArbinClient instance is connected to a server or not.

**Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

**Usage:**

```
myClient.IsConnected();
```

**Input:**

None

**Output:**

Return type	Description	Valid Values
boolean	Connection status	true - if the client is connected. false - if the client is not connected

### void ShutDown ()

- ✓ Shuts down the connection with the server.

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

#### Usage:

```
myClient.ShutDown();
```

#### Input:

None

#### Output:

Closes the connection.

## Initialization and Exiting

These functions are used to do set-up for issuing CTI commands and for receiving messages from the server in response to CTI commands.

### void Start ()

- ✓ This command is used to create a communication channel between client and server. It creates a new thread for receiving feedback messages sent by the server. This command is to be executed once at the start of the program.

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Following object needs to be defined,

```
ArbinControl myCtrl = new ArbinControl ();
```

Existing object of ArbinClient (e.g. myClient), which was created during connection.

#### Usage:

```
myCtrl.Start();
```

#### Input:

None

#### Output:

None

### void ListenSocketRecv (IArbinSocket socket)

- ✓ Starts listening on ports for feedback messages from the server.

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:**

```
myCtrl.ListenSocketRecv(myClient);
```

**Input:**

Parameter Name	Description	Data Type	Valid values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

**Output:**

None

## void Exit ()

- ✓ Shuts down the thread which receives messages from the cyclor. This Command is to be executed once at the end of program.

**Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:**

```
myCtrl.Exit();
```

**Input:**

None

**Output:**

None

## string GetCTIVersion()

- ✓ Get the current version number of ArbinCTI.dll

**Prerequisite:**

- ✓ ArbinCTI.DLL Version 1.0.0+

**Usage:**

```
myCtrl.GetCTIVersion();
```

**Input:**

None

**Output:**

Return type	Description	Valid Values
string	version number of ArbinCTI.dll	1.0.0



## Login Commands

User must login to server, to issue commands to the cycler. Following commands are used for logging-in.

### **bool PostLogicConnect (IArbinSocket socket, bool bSetKickOut);**

- ✓ Sets whether the logged-in user can be kicked out of the server.

#### **Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostLogicConnect(myClient, false);
```

#### **Input:**

Parameter Name	Description	Data Type	Valid values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
bSetKickOut	Kickout status	bool	true - can be kicked out false - cannot be kicked out

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void OnLogicConnectFeedBack](#) (ArbinCommandLogicConnectFeed cmd)

### **bool PostUserLogin (IArbinSocket socket, string strUser, string strPassword);**

- ✓ Log in to the server using the username and password created in CTI settings of server.

#### **Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostUserLogin(myClient, "myUsername", "myPassword");
```

**Inputs:**

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strUser	Username of the user that is allowed to log in through CTI	String	Actual username E.g. admin
strPassword	Password of the respective user	string	Actual password

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void OnUserLoginFeedBack](#) ([ArbinCommandLoginFeed cmd](#))

## Test Schedule Commands

These commands are used for assigning schedule on the channel or to change values of MetaVariables. It allows for assigning schedule or to change metavariable values for individual channel or all channels depending on the parameters passed.

### **bool PostModifySchedule(IArbinSocket socket, ModifyScheduleArgs args)**

✓ This command is used to modify some parameters of a schedule.

**Prerequisite:**

✓ MITS 10 4.2.19-patch and above versions are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

The schedule which is being used should be present in the work directory of server.

Object of ModifyScheduleArgs (e.g. args), along with required data in it.

**Usage:**

```
1 ModifyScheduleArgs args = new ModifyScheduleArgs();
2 args.ScheduleModifyInfos.Add(new ScheduleModifyInfo()
3 {
4     ScheduleName = "demo.sdx",
5     AuxDORequirement = new AuxChannelRequirementBase(true, 1),
6     AuxAORequirement = new AuxChannelRequirementBase(true, 1),
```

```

7   CANBMSRequirement = new AuxChannelRequirementBase(true, 1),
8   SMBRequirement = new AuxChannelRequirementBase(true, 1),
9
10  AuxVoltageRequirement = new AuxChannelRequirement(true, 12, new SafetyScope(-1.0f, 2.0f)),
11  AuxTemperatureRequirement = new AuxChannelRequirement(true, 1, new SafetyScope(-1.0f, 2.0f)),
12  AuxPressureRequirement = new AuxChannelRequirement(true, 1, new SafetyScope(-1.0f, 2.0f)),
13  AuxDIRequirement = new AuxChannelRequirement(true, 1, new SafetyScope(-1.0f, 2.0f)),
14  AuxExternalChargeRequirement = new AuxChannelRequirement(true, 1, new SafetyScope(-1.0f, 2.0f)),
15  AuxHumidityRequirement = new AuxChannelRequirement(true, 1, new SafetyScope(-2.0f, 2.0f)),
16
17  AuxSafelyRequirement = new AuxSafelyRequirement(true, 1, new SafetyScope(-1.0f, 1.0f), new
    SafetyScope(-2.0f, 3.0f), new SafetyScope(-3.0f, 4.0f)),
18  });
19
20  ctrl.PostModifySchedule(client, args);

```

#### Input:

Parameter Name	Description	Data type	Valid values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
args	Object of ModifyScheduleArgs	ModifyScheduleArgs	Refer: <a href="#">Annexure M</a> and <a href="#">Annexure O</a>

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void OnModifyScheduleFeedBack\(ArbinCommandModifyScheduleFeed cmd\);](#)

**bool PostAssignSchedule (IArbinSocket socket, string ScheduleName, String Barcode, float Capacity, float MVUD1, float MVUD2, float MVUD3, float MVUD4, [bool AllAssign = true], [int ChannelIndex = -1])**

- ✓ Assigns the schedule to channel(s).

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

But Parameters Barcode, Capacity, MVUD1, MVUD2, MVUD3 and MVUD4 are valid only for Pro7 and not for Pro8 and MITS 10.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

The schedule which is being assigned should be present in the work directory of server.

#### Usage:

```
myCtrl.PostAssignSchedule(myClient, "ScheduleFileName.sdx", "Barcode", 0.02f, 0, 0, 0, 0, false, 0);
```

```
myCtrl.PostAssignSchedule(myClient, "ScheduleFileName.sdx", "", 0.02f, 0, 0, 0, 0, false, 0);
```

```
myCtrl.PostAssignSchedule(myClient, "ScheduleFileName.sdx", "Barcode", 0.02f, 0, 0, 0, 0);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
ScheduleName	Name of Schedule	string	Actual name of schedule file
Barcode	Unique identifier for TestObject	string	Actual Barcode, it can be empty string also
Capacity	Nominal Capacity of Test object	float	Capacity of physical battery or any other object being tested
MVUD1,MVUD2,MVUD3,MVUD4	User can assign values to these User-defined meta variables	float	Any desired values
AllAssign	Whether to assign same schedule to all channels (Default value is true)	bool	true - assign the schedule to all channels  false - assign schedule only to the specified channel
ChannelIndex	Index of test channel (Index starts from 0 & default value is -1)	int	n - index of channel from 0 to (max no. of IV channels - 1)  -1 - will assign schedule to all channels if AllAssign is true else will set result object to "ASSIGN_INDEX_ERROR"



1. The nominal capacity is the amount of charge delivered by a fully charged battery under specified conditions of temperature and load.
2. MVUDs are mandatory to pass in the command. If user don't want to use MVUDs, pass 0.
3. If user don't pass last two parameters AllAssign and ChannelIndex. It will take default values and will assign same schedule to all channels.

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent  false - command not sent

The reply from the server for this command will be handled by [abstract void OnAssignScheduleFeedBack](#)  
(ArbinCommandAssignSchedule Feed cmd)

### **bool PostSetMetaVariable (IArbinSocket socket, int nChannel, int mvType, int mvMetaCode, int mvValueType, object mvValue)**

- ✓ Sends a command to change the value of MetaVariable. (One MetaVariable at a time.)

#### **Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_202101181+), Pro8 (TYMitsPro8\_201910311+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostSetMetaVariable(myClient, 8, 1, 52, 1, 0.02);
```

#### **Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
nChannel	Index of the channel	int	n - index of channel from 0 to (max no. of IV channels - 1)
mvType	Type of the MetaVariable	int	1 - Metavariable of IV channels (Other types are not supported)
mvMetaCode	Metacode of the Meta Variable	int	<a href="#">Annexure G : Meta Codes of MV_UDs</a>
mvValueType	Meta variable value type	int	1 - float
mvValue	Value of the Meta Variable	object	Any desired value

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnSetMVFeedBack](#)  
(ArbinCommandSetMetaVariableFeed cmd)

### **bool PostUpdateMetaVariableAdvanced (IArbinSocket socket, List<Meta VariableInfo> metaVariableList, out int error)**

- ✓ Sends a command to change the value of MetaVariables (Example: MV\_UD1, MV\_UD2)

**Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_202110131+), Pro8 (TYMitsPro8\_202110131+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<MetaVariableInfo> myVariableList = new List<MetaVariableInfo> ();
```

```
Int myErr;
```

**Usage:**

```
myCtrl.PostSetMetaVariableAdvanced(myClient, myVariableList, out myErr);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
metaVariable List	a list containing information for each MetaVariable	List<MetaVariableInfo>	<a href="#">Annexure B : List&lt;MetaVariableInfo&gt;</a>
error	Return error code	int	1:Means more than 160 data

**Output:**

Parameter Name	Description	Return type	Valid Values
error	Return error code	int	0 - command executed successfully  1 - more than 160 values in list were passed

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent  false - command not sent

The reply from server for this command will be handled by [abstract void OnUpdateMetaVariableAdvancedFeedBack](#) (ArbinCommand UpdateMetaVariableAdvancedFeed cmd)

**bool PostUpdateMetaVariableAdvancedEx (IArbinSocket socket, List<MetaVariableInfoEx> metaVariableList, out int error)**

- ✓ Sends a command to change the value of MetaVariables (Example: MV\_UD1, MV\_UD2)

**Prerequisite:**

✔ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In Development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<MetaVariableInfoEx> myVariableList = new List<MetaVariableInfoEx> ();
```

```
Int myErr;
```

#### Usage:

```
myCtrl.PostSetMetaVariableAdvanced(myClient, myVariableList, out myErr);
```

#### Input:

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
metaVariable List	a list containing information for each MetaVariable	List<MetaVariableInfoEx>	<a href="#">Annexure C : List&lt;MetaVariableInfoEx&gt;</a>
error	Return error code	int	-1: Exceeds the maximum nu mber of supported

#### Output:

Parameter Name	Description	Return type	Valid Values
error	Return error code	int	0 - command executed successfully  -1:Exceeds the maximum number of supported

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent  false - command not sent

The reply from server for this command will be handled by [abstract void OnUpdateMetaVariableAdvancedFeedBack](#) (ArbinCommand UpdateMetaVariableAdvancedFeed cmd)

#### public bool PostTimeSensitiveSetMV(IArbinSocket socket, TimeSensitiveSetMVArgs args)

- ✔ Command to update multiple MVUDs for multiple channels

#### Prerequisite:

✔ Only supported by MITS 10(5.1.26+ and also 4.1.8-patch+)



- An exception is thrown when the server(Mits) version is too old to support this command.
- Multiple Units can be updated in parallel, whereas the old method was time consuming while updating multiple units.



If TimeSensitiveSetMVArgs.TimeSensitiveSetMVChannel.IsDoLog is true, this successful call to PostTimeSensitiveSetMV will be logged to the database.

An entry will be made to IV\_Basic\_Table and IV\_Extended\_Table table as shown below

along(IV\_Extended\_Table's Data type and value can be found in [Annexure G: Meta Codes for MV\\_UDs](#))

Test ID	Channel ID	Date Time	Data Point	Test Time	Step Time	Cycle ID	Step ID	Current	Voltage	Charge Capacity	Discharge Capacity	Charge Energy	Discharge Energy	Data Flags	Limit ID	Data Flags2	Sub Step ID
0	1	17210129...	1	0	0	1	1	0	2.7	0	0	0	0	3670016	1	0	0

IV\_Basic\_Table

Test ID	Channel ID	Date Time	Data Point	Test Time	Step Time	Cycle ID	Step ID	Data Type	Data Value
0	1	17210129928949000	4	0	0	1	1	52	0.05

IV\_Extended\_Table

Size: A channel requires a data size of about 104 bytes, and an MV\_UD's requires a data size of about 72 bytes.

For example, updating 4 MV\_UDs of 1 channel requires 192 ( $1*104+4*72$ ) bytes of hard disc space space to store these data.

(If update at 5ms intervals, that will generate at least 6.3GB of data in a day)



- A successful call to PostTimeSensitiveSetMV means that OnTimeSensitiveSetMVFeedBack returns SUCCESS or SUCCESS\_NOTRUNNING.
- It is recommended to Sleep 10 ms after each successful call.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
TimeSensitiveSetMVArgs args = new TimeSensitiveSetMVArgs();
```

```
args.Timeout= 0.0f;
```

```
List<TimeSensitiveSetMV> mvudList = new List<TimeSensitiveSetMV>();
```

```
mvudList.Add(new TimeSensitiveSetMV(){MVUD = TimeSensitiveSetMV.EMVUD.MVUD1, Value = 0.05f});
```

```
mvudList.Add(new TimeSensitiveSetMV(){MVUD = TimeSensitiveSetMV.EMVUD.MVUD2, Value = 0.08f});
```

```
bool bDoLog = false;
```

```
TimeSensitiveSetMVArgs.TimeSensitiveSetMVChannel Channel1 = new  
TimeSensitiveSetMVArgs.TimeSensitiveSetMVChannel(0, mvudList, bDoLog)
```

```
TimeSensitiveSetMVArgs.TimeSensitiveSetMVChannel Channel2 = new  
TimeSensitiveSetMVArgs.TimeSensitiveSetMVChannel(1, mvudList, bDoLog)
```

```
args.Channels.Add(Channel1);
```


```
args.Channels.Add(Channel2);
```

#### Usage:

```
myCtrl.PostTimeSensitiveSetMV(myClient, args);
```

#### Inputs:



Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
args	Update which MVUD values for which channels	TimeSensitiveSetMVArgs	Cannot be null, reference <a href="#">TimeSensitiveSetMVArgs</a>
args.Timeout	If the interval between the next successful call to PostTimeSensitiveSetMV and the last successful call to PostTimeSensitiveSetMV is greater than the Timeout, the corresponding running channel is automatically stopped	float	The unit is seconds. Less than or equal to 0 is no timeout check.
bDoLog	Indicates whether this setting needs to be logged into the database.	bool	true, means need to log this setting to the database; false, means do not log to the database
	 Even when false, every Start/Resume will have all MV_UD data into the database, and every jump step will also have channel data into the database.		
mvudList	Up to 8 MV_UD variables can be set per channel.	List<TimeSensitiveSetMV >	Cannot be null, reference <a href="#">TimeSensitiveSetMVArgs</a>

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void OnTimeSensitiveSetMVFeedBack\(ArbinCommandTimeSensitiveSetMVFeed cmd\);](#)

**[bool PostGetMetaVariables\(IArbinSocket socket, List<ArbinCommandGetMetaVariablesFeed.MetaVariableInfo> metaVariableList, out int error\)](#)**

- ✓ Send command to get the value of some Meta Variables (example: MetaCode\_PV\_Voltage, MetaCode\_PV\_Current)

#### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_202203291+), Pro8 (TYMitsPro8\_202203301+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ArbinCommandGetMetaVariablesFeed.MetaVariableInfo> myVariableList = new  
List<ArbinCommandGetMetaVariablesFeed.MetaVariableInfo> ();
```

```
Int myErr;
```

**Usage:**

```
myCtrl. PostGetMetaVariables (myClient, myVariableList, out myErr);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
metaVariable List	a list containing information for each MetaVariable	List<ArbinCommandGetM etaVariablesFeed.MetaVari ableInfo>	MetaVariableInfo
error	Return error code	int	-1:Means more than 128 data

**Output:**

Parameter Name	Description	Return type	Valid Values
error	Return error code	int	-1: MetaVariableList data length exceeds 128; -2: 3 seconds timeout; -3: Failed to send

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetMetaVariablesFeedBack](#)  
(ArbinCommand ArbinCommandGetMetaVariablesFeed cmd)

**bool PostSetIntervalTimeLogData (IArbinSocket socket, float fIntervalTime)**

- ✓ Send command to set the time interval for MCU to upload Logged data

**Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_2022005071+), Pro8 (TYMitsPro8\_2022005071+) and MITS 10(In development) are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
float myIntervalTime;
```

**Usage:**

myCtrl.PostGetMetaVariables (myClient, myVariableList, out myErr);

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
fIntervalTime	time interval	float	0.005~3(s)

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnSetIntervalTimeLogDataFeedBack](#) (ArbinCommandSetIntervalTimeLogDataFeed cmd)

**bool PostApplyForUDPCommunication(IArbinSocket socket, ushort uUDPClientPort, string strUDPClientIP, List<CMetavarialeDataCodeApply> dataCodeApplys)**

- ✓ Send command to apply for UDP communication

**Prerequisite:**

- ✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

ushort myUDPClientPort = 9669;

string myUDPClientIP = "196.168.1.168";

List<CMetavarialeDataCodeApply> myVariableList = new List< CMetavarialeDataCodeApply> ();

**Usage:**

myCtrl.PostApplyForUDPCommunication(myClient, myUDPClientPort, myUDPClientIP, myVariableList);

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
uUDPClientPort	Socket port number of UDP client	ushort	1024~65535
strUDPClientIP	Socket IP of UDP client	string	"196.168.1.168"

dataCodeApplys	List of application communication	List<CMetavariableDataC odeApply>	<a href="#">Annexure D : List&lt;CMetavariableDataC odeApply&gt;</a>
----------------	-----------------------------------	--------------------------------------	--

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnApplyForUDPCommunicationFeedBack \(ArbinCommandApplyForUDPCommunicationFeed cmd\)](#)

**[bool PostUpdateParameters\(IArbinSocket socket, List<ushort> chanList, bool isForAllChans, SortedDictionary<ushort, double> parameters\)](#)**

✓ Send command to update parameters (e.g. TestObject's parameters)

#### Prerequisite:

✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ushort> myChannels = new List<ushort>();
```

```
bool bForAllChannels = false;
```

```
SortedDictionary<ushort, double> myParametersList = new SortedDictionary <ushort, double> ();
```

#### Usage:

```
myCtrl.PostUpdateParameters(myClient, myChannels, bForAllChannels, myParametersList);
```

#### Input:

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
chanList	List of channels to update	List<ushort>	n – index of channel from 0 to (max no. of IV channels – 1)
isForAllChans	true: Update all Channels; false: Update Channel in chanList	bool	false/true
parameters	Dictionary of update parameters	SortedDictionary<ushort, double>	EParameterDataType

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void UpdateParametersBack\(ArbinCommandUpdateParameterFeed cmd\)](#)

**Note:** Older versions of the clock, calling this interface, will be subject to two kinds of FeedBack: OnUpdateParametersBack and OnGetStatusExtendInformationBack; the latest version will only have OnUpdateParametersBack.

### **bool PostConvertToAnonymousOrNamedTO(IArbinSocket socket, bool to Anonymous, bool isForAll, List<ushort> channelList)**

- ✓ Send command to convert TestObject to anonymous or non-anonymous mode

#### **Prerequisite:**

✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

bool toAnonymous = false;

bool bForAll = false;

List<ushort> myChannels = new List<ushort>();

#### **Usage:**

myCtrl.PostConvertToAnonymousOrNamedTO(myClient, toAnonymous, bForAll, myChannels);

#### **Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
toAnonymous	true: Convert TestObject to anonymous; false: non-anonymous mode	bool	false/true
isForAll	true: Update all Channels; false: Update Channel in chanList	bool	false/true
channelList	List of channels to update	List<ushort>	n – index of channel from 0 to (max no. of IV channels – 1)

Return type	Description	Valid Values
-------------	-------------	--------------

boolean	Returns execution status	true - command sent false - command not sent
---------	--------------------------	---

The reply from server for this command will be handled by [abstract void](#)

[OnConvertToAnonymousOrNamedTOBack\(ArbinCommandConvertToAnonymousOrNamedTOFeed cmd\)](#)

**bool PostAssignBarcodeInfo(IArbinSocket socket, ArbinCommandAssignBarcodeInfoFeed.EChannelType channelType, List<ArbinCommandAssignBarcodeInfoFeed.ChannelBarcodeInfo> barcodeInfos)**

- ✓ Send a command to assign a barcode to channel

#### Prerequisite:

✓ Pro8(2.0.0+) and MITS 10(5.1.25+) versions, only Pro7 version is not supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ArbinCommandAssignBarcodeInfoFeed.ChannelBarcodeInfo> m_listBarcodeInfo = new
List<ArbinCommandAssignBarcodeInfoFeed.ChannelBarcodeInfo>();
```

```
ArbinCommandAssignBarcodeInfoFeed.EChannelType m_eChannelType =
ArbinCommandAssignBarcodeInfoFeed.EChannelType.IV;
```

#### Usage:

```
myCtrl.PostAssignBarcodeInfo(myClient, m_eChannelType, m_listBarcodeInfo);
```

#### Input:

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
channelType	Channel Type	ArbinCommandAssignBarcodeInfoFeed.EChannelType	
barcodeInfos	List of channels to update	List<ArbinCommandAssignBarcodeInfoFeed.ChannelBarcodeInfo>	

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void](#)

[OnAssignBarcodeInfoFeedBack\(ArbinCommandAssignBarcodeFeed cmd\)](#)

**bool PostGetBarcodeInfo(IArbinSocket socket, ArbinCommandGetBarcodeInfoFeed.EChannelType channelType, List<ArbinCommandGetBarcodeInfoFeed.GetChannelBarcodeInfo> barcodeInfos)**

✓ Send a command to get barcodes from service

**Prerequisite:**

✓ Pro8(2.0.0+) and MITS 10(5.1.25+) versions, only Pro7 version is not supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ArbinCommandGetBarcodeInfoFeed.GetChannelBarcodeInfo> m_listBarcodeInfo = new
List<ArbinCommandGetBarcodeInfoFeed.GetChannelBarcodeInfo>();
```

```
ArbinCommandGetBarcodeInfoFeed.EChannelType m_eChannelType =
ArbinCommandGetBarcodeInfoFeed.EChannelType.IV;
```

**Usage:**

```
myCtrl.PostGetBarcodeInfo(myClient, m_eChannelType, m_listBarcodeInfo);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
channelType	Channel Type	ArbinCommandGetBarcodeInfoFeed.EChannelType	
barcodeInfos	List of channels to update	List<ArbinCommandGetBarcodeInfoFeed.GetChannelBarcodeInfo>	

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetBarcodeInfoFeedBack\(ArbinCommandGetBarcodeInfoFeed cmd\)](#)

**bool PostGetMachineType(IArbinSocket socket)**

✓ Send a command to get machine type from service

**Prerequisite:**

✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:**

```
myCtrl.PostGetMachineType(myClient);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetMachineTypeFeedBack\(ArbinCommandGetMachineTypeFeed cmd\)](#)

**bool PostGetTrayStatus(IArbinSocket socket, List<ushort> trayIndexList, out int error)**

- ✓ Send a command to get tray status from service

**Prerequisite:**

- ✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
private List<ushort> m_trayIndexList
```

```
int m_nError
```

**Usage:**

```
myCtrl.PostGetTrayStatus(myClient, m_trayIndexList, out m_nError);
```

**Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
trayIndexList	List of tray global index	List<ushort>	
error	error code	int	-1:The number of exceeded and the excess are filtered.

Return type	Description	Valid Values
-------------	-------------	--------------



boolean	Returns execution status	true - command sent false - command not sent
---------	--------------------------	---

The reply from server for this command will be handled by [abstract void OnGetTrayStatusFeedBack\(ArbinCommandGetTrayStatusFeed cmd\)](#)

### **bool PostEngageTray(IArbinSocket socket, List<ArbinCommandEngageTrayFeed.EngageTrayInfo> engageTrayList, out int error)**

- ✓ Send a command to get engage tray from service

#### **Prerequisite:**

✓ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported (In development).

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

List<EngageTray> m\_engageTrayList

int m\_nError

#### **Usage:**

myCtrl.PostEngageTray(myClient, m\_engageTrayList, out m\_nError);

#### **Input:**

Parameter Name	Description	Data type	Valid values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
engageTrayList	List of tray global index	List<ArbinCommandEngageTrayFeed.EngageTrayInfo>	
error	error code	int	-1:The number of exceeded and the excess are filtered

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnEngageTrayFeedBack\(ArbinCommandEngageTrayFeed cmd\)](#)

## Channel Controls

These commands are used to control the channels. Like start/stop/resume a channel.

### **bool PostStartChannel (IArbinSocket socket, string TestName, List<ushort> Channels);**

- ✓ This command is used to start test on a channel or channels.

#### **Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ushort> myChannelList = new List<ushort> () {0,1,2,3,4,5,...n};
```

#### **Usage:**

```
myCtrl.PostStartChannel(myClient, "myTestName", myChannelList);
```

#### **Input:**

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
TestName	Descriptive name of a test	string	Any desired name
Channels	List of channels to start	List<ushort>	n - index of channel from 0 to (max no. of IV channels - 1)

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnStartChannelFeedBack](#) (ArbinCommandStartChannelFeed cmd)

### **bool PostStartChannelEx (IArbinSocket socket, List<StartResumeEx> resumeEx, string Creators, string Comments)**

- ✓ Starts the channels in the list, using the information stored in same list to initialize each channel's test information.

#### **Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.


```
List<StartResumeEx> myChannelList = new List<StartResumeEx> ();
```

**Usage:**

```
myCtrl.PostStartChannelEx(myClient, myChannelList, "creator", "comments");
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during Connection
resumeEx	a list containing information for each channel, in the same order as Channels.	List<StartResumeEx>	<a href="#">Annexure A : List&lt;StartResumeEx&gt;</a>
Creators	string containing the list of creators of this test/schedule	string	
Comments	string containing comments about this test/schedule	string	

 All channels start under the first item's TestNames string, but each channel begins using the corresponding Schedules string to determine the schedule to run

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnStartChannelFeedBack](#) ([ArbinCommandStartChannelFeed cmd](#))

**[bool PostStartChannelAdvanced\(IArbinSocket socket, StartChannelAdvancedArgs args\)](#)**

- ✓ Assigns the information passed in the StartChannelAdvancedArgs list (Schedule file, Test name, Test Object's parameters, barcode and Channel index) and then starts the channel.

**Prerequisite:**

- ✓ Versions of MITS 10 (MITS 10.4.2.2 Patch +) are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:**

```
myCtrl.PostStartChannelAdvanced(myClient, startChannelArgs);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
----------------	-------------	-----------	--------------

socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during Connection
startChannelArgs	Object of class: StartChannelAdvancedArg s	StartChannelAdvancedArg s	<a href="#">Annexure K : StartChannelAdvancedArg s</a>

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnStartChannelAdvancedFeedBack](#)  
(ArbinCommandStartChannelAdvancedFeed cmd)

### bool PostStopChannel (IArbinSocket socket, int nChannel, bool bStopAll)

- ✓ Send a command to the server to stop the selected channel or to stop all channels.

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### Usage:

myCtrl.PostStopChannel (myClient, 8, false)

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
nChannel	Index of Channel to stop	int	n - index of channel from 0 to (max no. of IV channels - 1)
bStopAll	Whether to stop all channels or the only one specified channel	boolean	true - to stop all channels false - to stop only one specified channel

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnStopChannelFeedBack](#) ([ArbinCommandStopChannelFeed cmd](#))

### **bool PostJumpChannel (IArbinSocket socket, int stepNum, int channelIndex)**

- ✓ Send a command to the server to jump the step in the schedule.

#### **Prerequisite:**

✓ Pro7 versions is not supported, only Pro8 (TYMitsPro8\_201909191+) and MITS 10 versions are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostJumpChannel(myClient, 3, 8);
```

#### **Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
nChannel	Index of Channel to jump	int	n – index of channel from 0 to (max no. of IV channels – 1)
stepNum	Step number need to jump to	int	1,2,...,n (n =last step number of the schedule)

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnJumpChannelFeedBack](#) ([ArbinCommandJumpChannelFeed cmd](#))

### **bool PostResumeChannel (IArbinSocket socket, bool AllResume, int Channel Index)**

- ✓ Send a command to resume all channels or a single specific channel.

#### **Prerequisite:**

✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostResumeChannel(myClient, false, 8);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
nChannel	Index of Channel to resume	int	n - index of channel from 0 to (max no. of IV channels - 1)
AllResume	Whether to resume all channels or the only one mentioned	boolean	true - to stop all channels false - to stop only one mentioned

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnResumeChannelFeedBack](#) ([ArbinCommandResumChannele Feed cmd](#))

**bool PostResumeChannelEx (IArbinSocket socket, List<StartResumeEx> resumeExs)**

- ✓ Send a resume command to the cycler to start the channels in Channels using the arguments passed in resumeExs.

**Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

List<StartResumeEx> myResumeList = new List<StartResumeEx> ();

**Usage:**

myCtrl.PostResumeChannelEx(myClient, myResumeList);

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
resumeExs	a list containing information for each channel, in the same order as Channels.	List<StartResumeEx>	<a href="#">Annexure A : List&lt;StartResumeEx&gt;</a>



All channels resume under the first item's TestName, but each channel resumes using the corresponding Schedules.

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnResumeChannelFeedBack](#) (ArbinCommandResumChannele Feed cmd)

**bool PostResumeChannelByMetaVariables(IArbinSocket socket, List<StartResumeEx> resumeExs)**

- ✓ Send a resume command to the cycler to start the channels in Channels using the arguments passed in resumeExs.

**Prerequisite:**

✔ Pro7 and Pro8 versions are not supported, only MITS 10 versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

List<StartResumeEx> myResumeList = new List<StartResumeEx> ();

**Usage:**

myCtrl.PostResumeChannelEx(myClient, myResumeList);

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
resumeExs	a list containing information for each channel, in the same order as Channels.	List<StartResumeEx>	<a href="#">Annexure A : List&lt;StartResumeEx&gt;</a>

❗ All channels resume under the first item's TestName, but each channel resumes using the corresponding Schedules.

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnResumeChannelFeedBack](#) (ArbinCommandResumChannele Feed cmd)

## bool PostContinueChannel(IArbinSocket socket, List<ushort> Channels)

- ✓ This command is used to continue test on channels.

### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_20210901+), Pro8 (TYMitsPro8\_20200901+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ushort> myChannelList = new List<ushort> ();
```

### Usage:

```
myCtrl.PostContinueChannel(myClient, myChannelList);
```

### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
Channels	List of channels to continue.	List<ushort>	n – index of channel from 0 to (max no. of IV channels – 1)

### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnContinueChannelFeedBack\(ArbinCommandContinueChannelFeed cmd\)](#)

## File Operation Commands

These commands are used for browsing, sending, and retrieving the files from the server.

## bool PostBrowseDirectory (IArbinSocket socket, string strPath)

- ✓ Sends a command to the server to retrieve a list of files stored at strPath if the path exists.

### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
string myPath = ".....\\WinDaq\\Work; (path of work directory of the server)
```

### Usage:



myCtrl.PostBrowseDirectory (myClient, myPath)

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strPath	Path of the directory you want to browse.	string	work folder of server

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnBrowseDirectoryBack](#) ([ArbinCommandBrowseDirectoryFeed cmd](#))

### **bool PostDeleteFolder (IArbinSocket socket, string strPath)**

- ✓ Send a command to the server to delete the folder specified by strPath.

**Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

string myPath = “.....WinDaq\\work\\myFolder”

**Usage:**

myCtrl.PostDeleteFolder (myClient, myPath);

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strPath	Folder name along with its path	string	

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnDeleteFolderBack\(ArbinCommandDeleteFolderFeed cmd\)](#)

### **bool PostDownloadFile (IArbinSocket socket, string strPath, double time)**

- ✓ Send a request to the server to send the file to the client.

#### **Prerequisite:**

✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

string myPath = File name along with actual path where it exists on server.

#### **Usage:**

myCtrl.PostDownloadFile (myClient, myPath, 0)

#### **Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
strPath	File name including path of the file	string	myPath
time	Timeout of the operation	double	You can give any value in seconds (0 - infinite)

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnDownloadFileBack\(ArbinCommandDownloadFileFeed cmd\)](#)

### **bool PostNewFolder (IArbinSocket socket, string strPath)**

- ✓ Send a command to the server to create a folder at strPath.

#### **Prerequisite:**

✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

string myPath = Folder name along with actual path where you want to create it.

**Usage:**

myCtrl.PostPostNewFolder (socket, myPath);

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strPath	Folder name along with actual path where you want to create it.	string	C:\\...\\Windaq\\Work\\my Folder

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnNewFolderBack\(ArbinCommandNewFolderFeed cmd\)](#)

**bool PostNewOrDelete (IArbinSocket socket, string strPath, ArbinCommand NewOrDeleteFeed.NEW\_OR\_DELETE\_TYPE uType)**

- ✓ Send a command to the server to delete the file specified by strPath or create a folder at strPath.

**Prerequisite:**

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

string myPath = File name/Folder name along with actual path where you want to create/delete file or folder.

- ✓ Send a command to the server to delete the file specified by strPath or create a folder at strPath.

**Usage:**

myCtrl.PostNewOrDelete(myClient, myPath, ArbinCommandNewOrDeleteFeed.NEW\_OR\_DELETE\_TYPE.CTI\_NEW);

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

strPath	File/Folder name and actual path where you want to create/delete file or folder	string	C:\\...\\Windaq\\Work\\myFile
uType	Whether to create folder or delete file	ArbinCommandNewOrDeleteFeed.NEW_OR_DELETE_TYPE	CTI.NEW - for creating folder CTI.DELETE - for deleting folder

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by `abstract void OnNewOrDeleteBack(ArbinCommandNewOrDeleteFeed cmd)`

### **bool PostUploadFile (IArbinSocket socket, string strPath, byte[] Filedata, double time, uint uGeneralPackage, uint PackageIndex)**

- ✓ Sends a command to the server to allow CTI to begin uploading a file to the server. The file data is passed as a byte array.

#### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_201909261+), Pro8 (TYMitsPro8\_201906141+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
var myStream = new FileStream (myFileName, FileMode.OpenOrCreate)
```

*(myFileName is the file which you want to upload).*

```
stream.Position = 0;
```

```
long fileSize = stream.Length;
```

```
byte [] myFileData = converted file data in bytes to be copied in this array.
```

```
totalPackets = (int) ((fileSize % packByteSize) != 0? (fileSize / packByteSize) + 1: (fileSize / packByteSize))
```

```
lastPack = totalPackets - 1; (packageIndex is 0 to lastPack)
```

#### Usage:

```
myCtrl.PostUploadFile (myClient, myFileName, myFileData, 0, totalPackets, packageIndex);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

strPath	Absolute file path at the destination	string	e.g. "C:/ArbinSoftware/MITS/Work"
Filedata	File data in the form of bytes	byte []	
time	Timeout (Seconds)	int	0 - infinite
uGeneralPackage	Total number of packets	uint	
PackageIndex	Index of the package	uint	

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnUploadFileBack\(ArbinCommandUploadFileFeed cmd\)](#)

**bool PostUploadFile(IArbinSocket socket, string strRemoteRelativeFileName, string strLocalFullFileName, bool bOverride, ArbinCommandUploadFile Feed.CUploadFileResult.AsyncCallback uploadFileResultCallback)**

- ✓ It is an overload to above PostUploadFile(). It Sends a command to the server to allow CTI to begin uploading a file to the server. It's used in Advanced Assign control.

#### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_202207071+), Pro8 (TYMitsPro8\_202207071+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### Usage:

```
myCtrl.PostUploadFile(client, "Work/dvfb_Channel_1_Wb_1.xlsx",
"C:/ArbinResultData/dvfb_2022_05_19_135108/dvfb_Channel_1_Wb_1.xlsx", false, TransferUploadThreadMethod);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strRemoteRelativeFileName	Relative path of the remote server where you want to upload file	string	C:/ArbinSoftware/MITS_Pro <b>OR</b> C:/ArbinSoftware/MITS_Pro /Work
strLocalFullFileName	Local path of the file to be uploaded	string	Any accessible path

bOverride	If file with same name is available on the server whether to override it or not	boolean	True – Override False – Don't Override
uploadFileResultCallback	Call back event	ArbinCommandUpLoadFileFeed.CUpLoadFileResult.AsyncCallback	TransferUploadThreadMethod

- i** 1. uploadFileResultCallback - During the upload process, you can get the percentage of upload progress through ProgressRate. You can also use the IsCancelUploadFile flag to cancel the file upload. You can know the execution status of the PostUpLoadFile interface according to the ResultCode.
2. If just a file name is passed in strRemoteRelativeFileName without path, it will by default take the path based on the file extension. For .sdx or .sdu file path will be **C:/ArbinSoftware/MITS\_PRO/work** and for rest of the extensions it will be **C:/ArbinSoftware/MITS\_PRO**.

#### TransferUploadThreadMethod – Example use

```

1 private void TransferUploadThreadMethod(ref ArbinCommandUpLoadFileFeed.CUpLoadFileResult result)
2 {
3     Console.WriteLine(result.ResultCode);
4     Console.WriteLine(result.ProgressRate*100);
5 }

```

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnUpLoadFileBack\(ArbinCommandUpLoadFileFeed cmd\)](#)

### bool PostCheckFileEx(IArbinSocket socket, string strRemoteRelativeFile Name, byte[] MD5);

- ✓ Checks whether the file is available on remote server, if it exists checks whether the MD5 is consistent.

#### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_202207071+), Pro8 (TYMitsPro8\_202207071+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### Usage:

```
myCtrl.PostCheckFileEx(client, "Default+Default.sdx", fileMD5);
```

```
myCtrl.PostCheckFileEx(client, "TestData.xlsx", fileMD5);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
----------------	-------------	-----------	--------------

socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
strRemoteRelativeFileName	Relative full path of the remote server where you want to check file existence	string	
MD5	MD5 code of local file to check with	Byte[]	

- i** 1. If the remote end finds the file, the MD5 code of the file is paired to determine whether the contents of the local file and the remote file are consistent.
2. For .sdx and .sdu file it will check existence in **C:/ArbinSoftware/MITS\_PRO/Work** directory and for another file in **C:/ArbinSoftware/MITS\_PRO**.

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnCheckFileExBack\(ArbinCommandCheckFileExFeed cmd\);](#)

### **bool PostAssignFile(IArbinSocket socket, string fileName, bool bAssignAll, ArbinCommandAssignFileFeed.EFileKind eFileKind, List<ushort> chanList)**

- ✓ It is an advanced API for PostAssignSchedule(). Using this one can assign any file to the channel like schedule file, CAN file, BMS file etc.

#### Prerequisite:

- ✓** All versions of Pro7 (TYMitsPro7\_202207071+), Pro8 (TYMitsPro8\_202207071+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

List<ushort> myChannellist = new List<ushort> () {0,1,2,3,4,5,...n};

#### Usage:

```
myCtrl.PostAssignFile(client, "ccc.sdu", true, ArbinCommandAssignFileFeed. EFileKind.Schedule, Chnl);
```

```
myCtrl.PostAssignFile(client, "Test001.can", false, ArbinCommandAssignFileFeed. EFileKind.CANBMS, Chnl);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
fileName	Name of the file to assign	String	.sdx, .sdu, .smb, .can

bAssignAll	Whether to assign the file to all channels or the only to specified ones.	Boolean	True – Assign to all False – only to specified
eFileKind	Type of the file to assign	ArbinCommandAssignFileFeed.EFileKind	Schedule , CANBMS , SMB , Chart
chanList	List of the channels to assign file	List<ushort>	n – index of channel from 0 to (max no. of IV channels – 1)

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnAssignFileFeedBack\(ArbinCommandAssignFileFeed cmd\)](#)

## Messaging Commands

These commands are used to send text messages to the cyclor. The cyclor will display the message in a dialog box.

### bool PostSendMsgToCTI (IArbinSocket socket, string msg)

- ✓ It opens a dialog window on the server with the formatted text provided by string msg. Message must be provided in rich-text format. Otherwise, garbage will be displayed. It allows for changing font, size, and color of the text.

#### Prerequisite:

- ✓ Pro7 and MITS 10 versions is not supported, only Pro8 (TYMitsPro8\_201909191+) versions is supported

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

string myMsg = Actual message to be sent to cyclor PC.

#### Usage:

```
myCtrl.PostSendMsgToCTI (myClient, myMsg);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
msg	The message to send to the server	string	Must be provided in a rich-text format. Otherwise, garbage will be displayed.

#### Output:



Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnSendMsgToCTIBack](#)  
(ArbinCommandSendMsgToCTIFeed cmd)

## Calibration Commands

These commands are used to start Calibration.

### **bool PostStartAutomaticCalibration (IArbinSocket socket)**

- ✓ Send a command to enable automatic calibration.

#### **Prerequisite:**

- ✓ Pro7 and MITS 10 versions is not supported, only Pro8 (TYMitsPro8\_201909191+) versions is supported

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

myCtrl. PostStartAutomaticCalibration (myClient);

#### **Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract](#)  
[OnStartAutomaticCalibrationBack](#)(ArbinCommandStartAutomaticCalibrationFeed cmd)

## Request Information Commands

These commands are used to request information of cycler/Channels from the server.

### **bool PostGetMappingAux (IArbinSocket socket, GetMappingAuxArgs args)**

- ✓ Send a command to the server to get the mapping details.

**Prerequisite:**

✔ MITS 10 4.2.19-patch and above are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

Object of GetMappingAuxArgs (e.g. args), created new here.

**Usage:**

```
GetMappingAuxArgs args = new GetMappingAuxArgs();
```

```
myCtrl.PostGetMappingAux(myClient, args);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
args	Object of class GetMappingAuxArgs	GetMappingAuxArgs	NA (Refer : <a href="#">Annexure M</a> )

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by `abstract void OnGetMappingAuxFeedBack(ArbinCommandGetMappingAuxFeed cmd);`

**bool PostGetChannelsData (IArbinSocket socket, uint NeedType, short OnlyGet ChannelNumber, ArbinCommandGetChannelFeed.GET\_ CHANNEL\_ TYPE GetChannelType)**

✓ Send a command to the server to get the status of cyclor's channels

**Prerequisite:**

✔ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:****//For all data**

```
myCtrl.PostGetChannelsData (myClient, 1792, myChannel, myChannelType);
```

**//For AUX data**

```
myCtrl.PostGetChannelsData (myClient, 1024 , myChannel, myChannelType);
```

**//For SMB data**


```
myCtrl.PostGetChannelsData (myClient, 512 , myChannel, myChannelType);
```

### //For BMS data

```
myCtrl.PostGetChannelsData (myClient, 256 , myChannel, myChannelType);
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
NeedType	cycler should give the information on the chosen types.	uint	Given below in note.
OnlyGetChannelNumber	Only retrieve the status of the specified channel	short	n - index of channel from 0 to (max no. of IV channels - 1)
GetChannelType	Gets the status of only the type of channel specified	ArbinCommandGetChannelFeed.GET_CHANNEL_TYPE	ALLCHANNEL, RUNNING, UNSAFE

 Possible valid values for “uint NeedType” are THIRD\_PART\_GET\_CHANNELS\_INFO\_NEED\_TYPE\_BMS = **0x100**, THIRD\_PART\_GET\_CHANNELS\_INFO\_NEED\_TYPE\_SMB = **0x200**, and THIRD\_PART\_GET\_CHANNELS\_INFO\_NEED\_TYPE\_AUX = **0x400**, and values obtained by OR-ing these together.

#### Output:


Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetChannelsDataFeedBack](#) ([ArbinCommandGetChannel DataFeed cmd](#))

### **bool PostGetChannelsDataSPTT(IArbinSocket socket, short OnlyGetChannelNumber= -1)**

- ✓ Send a command to the server to get the SPTT data of cycler's channels

#### Prerequisite:

 (For MITS 10, In development)

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### Usage:

##### //For all Channels

```
myCtrl.PostGetChannelsDataMminimalistMode (myClient);
```

##### //For a particular channel (eg:6)

```
myCtrl.PostGetChannelsDataMminimalistMode (myClient, 5);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
OnlyGetChannelNumber	Only retrieve the status of the specified channel	short	n – index of channel from 0 to (max no. of IV channels – 1)

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetChannelsDataSPTTFeedBack \(ArbinCommandGetChannelDataSPTTFeed cmd\)](#)

**bool PostGetChannelsDataMminimalistMode(IArbinSocket socket, short onlyGetChannelNumber = -1)**

- ✓ Send a command to the server to get the status of cycler's channels. (Only Current and the Voltage).

**Prerequisite:**

 All versions of Pro7 (TYMitsPro7\_202203291+), Pro8 (TYMitsPro8\_202203301+) and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:****//For all Channels**

```
myCtrl.PostGetChannelsDataMminimalistMode (myClient);
```

**//For a particular channel (eg:6)**

```
myCtrl.PostGetChannelsDataMminimalistMode (myClient, 5);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
OnlyGetChannelNumber	Only retrieve the status of the specified channel	short	n – index of channel from 0 to (max no. of IV channels – 1)

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetChannelsDataMinimalistModeFeedBack\(ArbinCommand GetChannelDataMinimalistModeFeed cmd\);](#)

**bool PostGetChannelsDataSimpleMode(IArbinSocket socket, short OnlyGet ChannelNumber = -1, ArbinCommandGetChannelDataSimpleModeFeed. GET\_CHANNEL\_TYPE GetChannelType = ArbinCommandGetChannel DataSimpleModeFeed.GET\_CHANNEL\_TYPE.ALLCHANNEL)**

- ✓ Send a command to the server to get the status of cyclers's channels according to their state. (All, Running, Unsafe)

#### Prerequisite:

- ✓ All versions of Pro7 (TYMitsPro7\_202203291+), Pro8 (TYMitsPro8\_202203301+) and MITS 10(In development) are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### Usage:

##### //For all Channels

```
myCtrl.PostGetChannelsDataSimpleMode (myClient, , ArbinCommandGetChannel
DataSimpleModeFeed.GET_CHANNEL_TYPE.ALLCHANNEL);
```

##### //For all Running Channels

```
myCtrl.PostGetChannelsDataSimpleMode (myClient, , ArbinCommandGetChannel
DataSimpleModeFeed.GET_CHANNEL_TYPE.RUNNING);
```

##### //For a particular channel (eg:6)

```
myCtrl.PostGetChannelsDataMminimalistMode (myClient, 5, ArbinCommandGet
ChannelDataSimpleModeFeed.GET_CHANNEL_TYPE.ALLCHANNEL );
```

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
OnlyGet ChannelNumber	Only retrieve the status of the specified channel	short	n – index of channel from 0 to (max no. of IV channels – 1)
GetChannelType	Gets status of the type of channels specified	ArbinCommandGetChannelFeed.GET_CHANNEL_TYPE	ALLCHANNEL, RUNNING, UNSAFE

#### Output:

Return type	Description	Valid Values
-------------	-------------	--------------

boolean	Returns execution status	true - command sent false - command not sent
---------	--------------------------	---

The reply from server for this command will be handled by [abstract void OnGetChannelsDataSimpleModeFeedBack\(ArbinCommand GetChannelDataSimpleModeFeed cmd\);](#)

### bool PostGetResumeData (IArbinSocket socket, List<ushort> channels)

- ✓ Send a request to the server to retrieve the resume status of the channels.

#### Prerequisite:

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

List<ushort> myChannels = new list<ushort> ();

#### Usage:

myCtrl.PostGetResumeData (myClient, myChannels);

#### Input:

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
channels	List of channels to retrieve resume status information from the cyclor	List<ushort>	n – index of channel from 0 to (max no. of IV channels – 1)

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetResumeDataBack \(ArbinCommandGetResumeDataFeed cmd\)](#)

### bool PostGetSerialNumber(IArbinSocket socket)

- ✓ Send a command to the server to get the serial number of a cyclor.

#### Prerequisite:

- ✓ Pro7 and MITS 10 versions are not supported, only Pro8 (TYMitsPro8\_201909191+) versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

**Usage:**

```
myCtrl.PostGetSerialNumber (myClient);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetSerialNumberFeedBack \(ArbinCommandGetSerial NumberFeed cmd\);](#)

**bool PostGetStartData (IArbinSocket socket, List<ushort> channels)**

- ✓ Sends a command to the server to return the information about the schedule assigned to each channel in the channels list.

**Prerequisite:**

- ✓ All versions of Pro7, Pro8 and MITS 10 are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
List<ushort> myChannels = new list<ushort> ();
```

**Usage:**

```
myCtrl.PostGetStartData (myClient, myChannels);
```

**Input:**

Parameter Name	Description	Data Type	Valid Values
Socket	Object of class: ArbinClient	ArbinClient	object of ArbinClient created during connection
channels	0 indexed channels to get start status data	List<ushort>	n - index of channel from 0 to (max no. of IV channels - 1)

**Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from server for this command will be handled by [abstract void OnGetStartDataBack](#)  
(ArbinCommandGetStartDataFeed cmd)

### **bool PostGetServerSoftwareVersionNumber(IArbinSocket socket);**

- ✓ Get the version number of the Cyclor for this socket.

#### **Prerequisite:**

- ✓ Only supported by MITS 10( 3.0.0+ )

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

#### **Usage:**

```
myCtrl.PostGetServerSoftwareVersionNumber(myClient);
```

#### **Inputs:**

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection

#### **Output:**

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void](#)  
[OnGetServerSoftwareVersionNumberFeedBack](#)(ArbinCommandGetServerSoftwareVersionNumberFeed cmd)

### **bool PostGetStringLimitLength(IArbinSocket socket, uint NeedType = ((uint)ArbinCommandGetStringLimitLengthFeed.ECTIStrngLimitLengthType.TestName))**

- ✓ Get string limit length command send by Client

#### **Prerequisite:**

- ✓ All versions of Pro7( TYMitsPro7\_20230529+ ), Pro8 ( TYMitsPro8\_202300613+ )and MITS 10( 3.0.0+ ) are supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
uint NeedType = ((uint)ArbinCommandGetStringLimitLengthFeed.ECTIStrngLimitLengthType.TestName);
```

#### **Usage:**

```
myCtrl.PostGetServerSoftwareVersionNumber(myClient);
```

#### **Inputs:**

Parameter Name	Description	Data Type	Valid Values
----------------	-------------	-----------	--------------



socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
NeedType	Get the maximum length of a string of what type	int	ArbinCommandGetStringLimitLengthFeed.ECTIStringLimitLengthType

#### Output:

Return type	Description	Valid Values
boolean	Returns execution status	true - command sent false - command not sent

The reply from the server for this command will be handled by [abstract void OnGetStringLimitLengthFeedBack\(ArbinCommandGetStringLimitLengthFeed cmd\)](#)

### Task<GetUDSSignalValueResponse> SendGetUDSSignalValue(IArbinSocket socket, GetUDSSignalValueRequest request)

✓ Signal Values of the UDS Message returned by the requesting server

#### Prerequisite:

✓ Only MITS 10(5.0.0 Release+) versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
var request = new GetUDSSignalValueRequest();
    request.MessageName = "test1";
    request.GlobalChannelIndex = 0;
    request.Source = Common.UDSSignalSource.Response;
    request.GetSignalValues.Add("foo_a");
    request.GetSignalValues.Add("foo_b");
```

#### Usage:

```
var response = myCtrl.SendGetUDSSignalValue(myClient, request);
```

#### Inputs:

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
request	Request to get the Signal Values of a UDS Message for an IV channel (SignalSource: Response/Request)	GetUDSSignalValueRequest	Cannot be null

#### Output:

Return type	Description	Valid Values
Task<GetUDSSignalValueResponse>	Return results asynchronously	No exception thrown - command sent  Throw exception - command not sent

### Task<SetRequestUDSSignalValueResponse> SendSetUDSSignalValue(IArbinSocket socket, SetRequestUDSSignalValueRequest request)

- ✓ Setting the signal values for UDS message

#### Prerequisite:

- ✓ Only MITS 10(5.0.0 Release+) versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
var request = new SetRequestUDSSignalValueRequest();
    request.MessageName = "test1";
    request.GlobalChannelIndex = 0;
    request.SetSignalValuesInRequest.Add(new Common.NameValuePair<string> {Name = "foo__a", Value = "0xaa"});
    request.SetSignalValuesInRequest.Add(new Common.NameValuePair<string> {Name = "foo__b", Value = "0xbb"});
```

#### Usage:

```
var response = myCtrl.SendSetUDSSignalValue(myClient, request);
```

#### Inputs:

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
request	Setting the signal values for UDS message for an IV channel	SetRequestUDSSignalValueRequest	Cannot be null

#### Output:

Return type	Description	Valid Values
Task<SetRequestUDSSignalValueResponse>	Return results asynchronously	No exception thrown - command sent  Throw exception - command not sent

## Task<SendUDSMessageResponse> SendSendUDSMessage(IArbinSocket socket, SendUDSMessageRequest request)

✓ Send UDS Message to Firmware

### Prerequisite:

- ✓ Only MITS 10(5.0.0 Release+) versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
var request = new SendUDSMessageRequest();
    request.MessageName = "test1";
    request.GlobalChannelIndex = 0;
```

### Usage:

```
var response = myCtrl.SendSendUDSMessage(myClient, request);
```

### Inputs:

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
request	Send an IV channel UDS Message to Firmware	SendUDSMessageRequest	Cannot be null

### Output:

Return type	Description	Valid Values
Task<SendUDSMessageResponse>	Return results asynchronously	No exception thrown - command sent  Throw exception - command not sent

## Task<ExecUDSUnlockResponse> SendExecUDSUnlock(IArbinSocket socket, ExecUDSUnlockRequest request)

✓ Use the unlock function written by Arbin to send Security Access

### Prerequisite:

- ✓ Only MITS 10(5.0.0 Release+) versions is supported.

Existing object of ArbinClient (e.g. myClient), which was created during connection.

Existing object of ArbinControl (e.g. myCtrl), which was created during initialization.

```
var request = new ExecUDSUnlockRequest();
    request.Level= 0x01;//Odd number
    request.GlobalChannelIndex = 0;
```

### Usage:

```
var response = myCtrl.SendExecUDSUnlock(myClient, request);
```

**Inputs:**

Parameter Name	Description	Data Type	Valid Values
socket	Object of class ArbinClient	ArbinClient	object of ArbinClient created during connection
request	Send an IV channel UDS Message to Firmware	ExecUDSUnlockRequest	Cannot be null

**Output:**

Return type	Description	Valid Values
Task<ExecUDSUnlockResponse>	Return results asynchronously	No exception thrown - command sent  Throw exception - command not sent