Mass Software Upgrade (MSU)

Schneider Department: xxxxxx

Project Code: xxxxxx

Document Identification: AOCI/PMP/DE/T/6

Deliverable Name	MSU Server API Reference Guide	Stage Gate	
		OPEN	☐
		SELECT	☐
		DO	☐
		IMPLEMENT	☐
		PRODUCE	☐
		SELL	☐
		CLOSE	☐

Status	Draft ☒	In Review ☐	Official ☐

Roles	Function	Name
Authors	**Associate Lead – Software**	**Chetan N**
Reviewers		
Approvers		

Table of Contents

1 MSU OVERVIEW

Mass Software Upgrade [MSU] is a reliable network protocol for carrying out simultaneous firmware/software upgrade process of a large number of systems. It uses a combination of multicast and unicast datagram protocols for data transfer. Reliability is achieved by selective re-transmission of the desired portion of the data upon request from the systems participating in the upgrade process.

The objectives of Mass Software Upgrade process are:

- To simultaneously upgrade firmware/software/Application/Files on many systems.
- To encourage indirect or implicit (via programs) use of remote computers.
- To transfer data reliably and efficiently even on high traffic network.
- To Automatic recovery of upgrade process even in cases of network re-establishment.

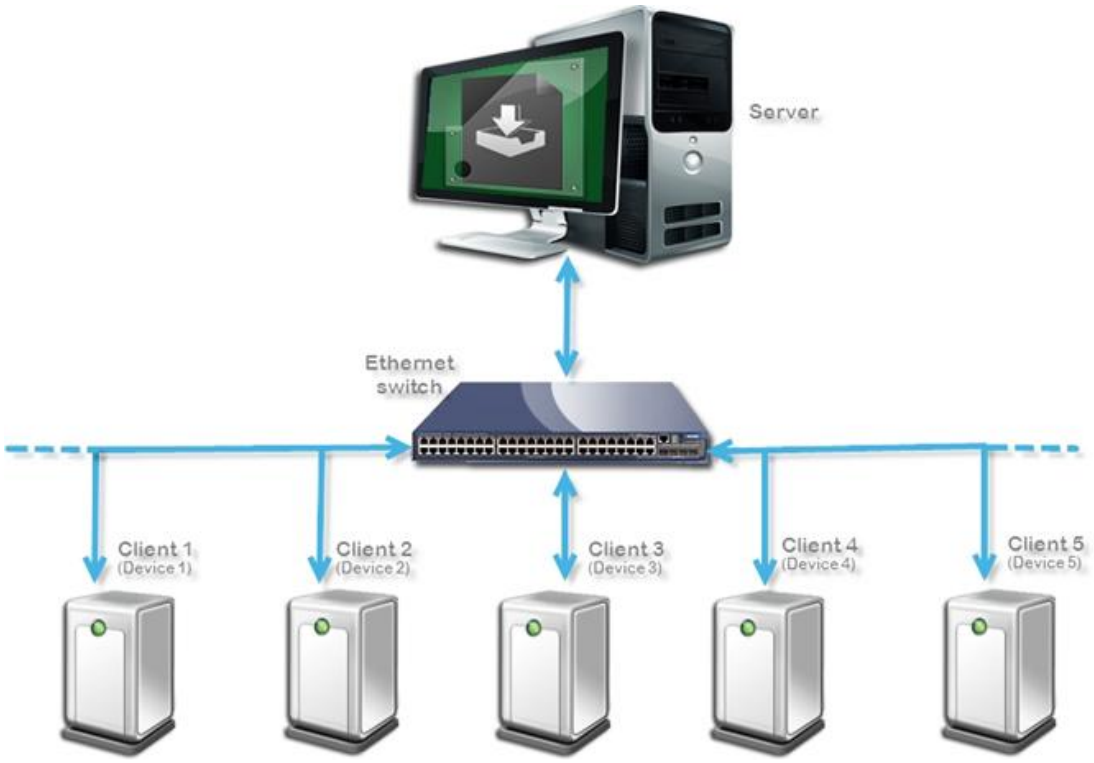


Figure 1 : MSU Solution Architecture

2 MSU SERVER API LIBRARY

MSU Server API Library contains the APIs for the MSU Server Operations like transactions with the device, file properties calculation etc.

MSU Server API Library is written in c# language under the .Net framework 4.0. MSU Server API Library is exposed as an assembly to other applications. This API library can be used with any other application which is developed under .Net framework.

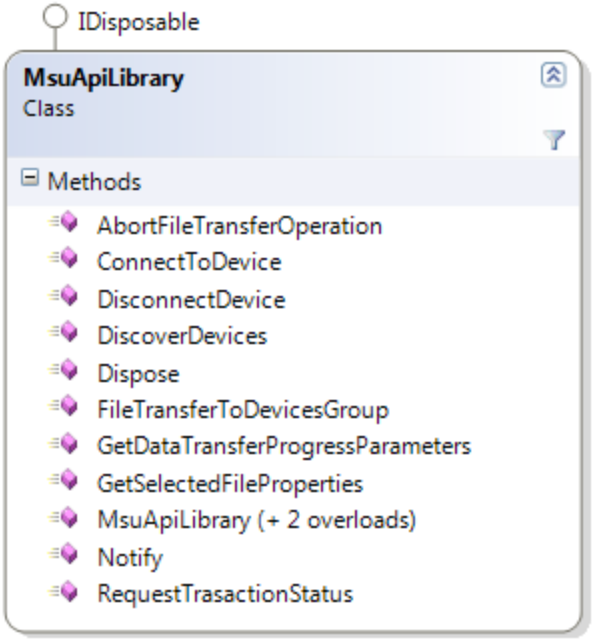
2.1 References:

Index	Document Title	Document ID	Version
1	MSU Offer Requirements		V 01.0
2	MSU Protocol Specification		V 01.0
3	MSU Server Interface Data Definition		V 01.0

Note: *The Message frames class types are defined in a separate assembly. For ready reference of the MSU Message frames data types refer MSU Message Frames Definition Document.*

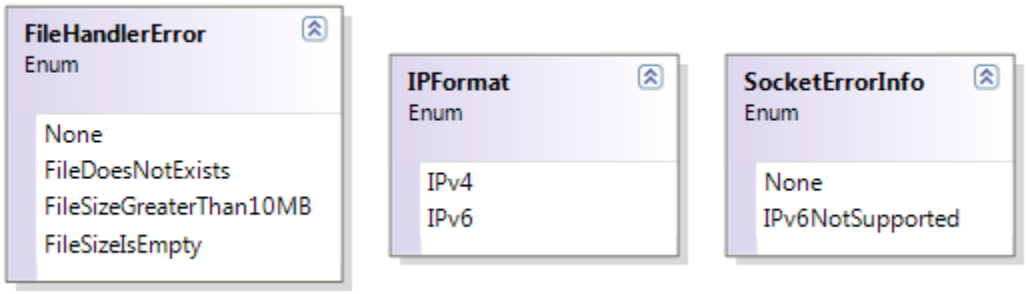
3 CLASS DIAGRAMS

3.1 API Class

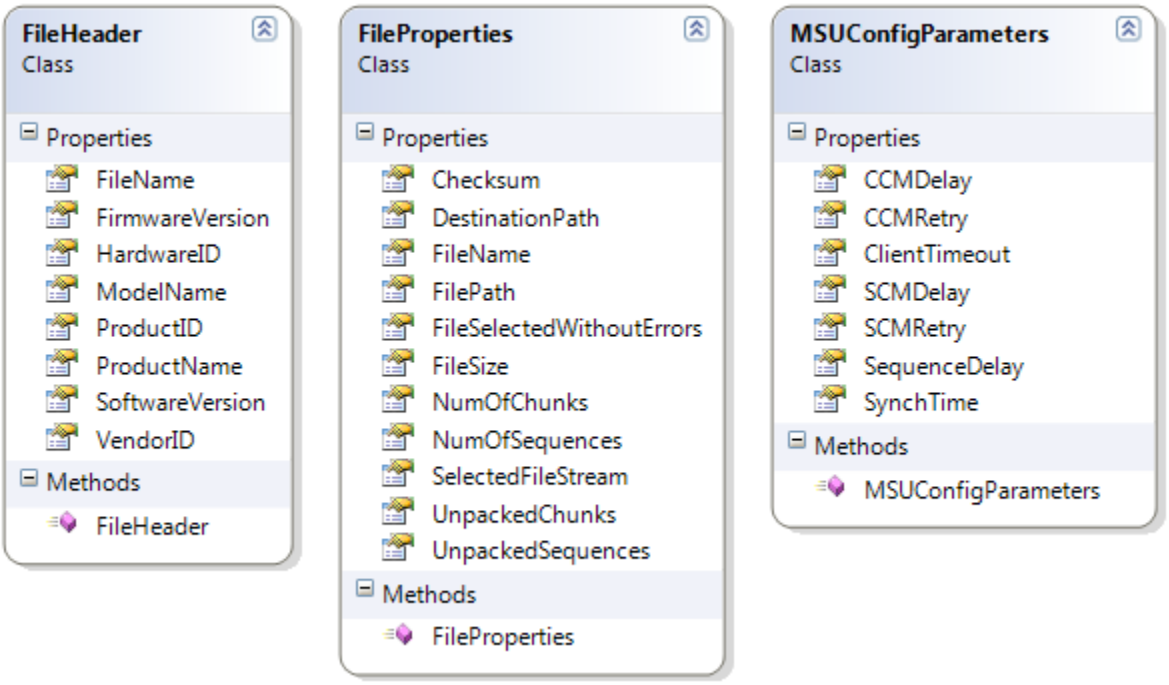


3.2 Helper Types

3.2.1 Enumerations



3.2.2 Classes


FileHeader
Class
Properties
FileName
FirmwareVersion
HardwareID
ModelName
ProductID
ProductName
SoftwareVersion
VendorID
Methods
FileHeader
FileProperties
Class
Properties
Checksum
DestinationPath
FileName
FilePath
FileSelectedWithoutErrors
FileSize
NumOfChunks
NumOfSequences
SelectedFileStream
UnpackedChunks
UnpackedSequences
Methods
FileProperties
MSUConfigParameters
Class
Properties
CCMDelay
CCMRetry
ClientTimeout
SCMDelay
SCMRetry
SequenceDelay
SynchTime
Methods
MSUConfigParameters

4 MSU SERVER APIS

4.1 Discover Devices

Scans for the devices over the network and responds with the list of devices responded for Scan operation.

```
public SocketErrorInfo DiscoverDevices(string localIpAddress,
                                       bool isRange,
                                       string startRange,
                                       string endRange,
                                       out ObservableCollection<MSUIam>
                                       devicesDiscovered);
```

Namespace:

MSUServer.MSUApiLibrary

Parameters:

localIPAddress

Type: String

Local IP Address from which the Client-server Transaction should happen.

isRange

Type: Boolean

Whether scan operation to be done in the range given.

startRange

Type: String

Start Range of the IP Address for scan.

endRange

Type: String

End Range of the IP Address for scan.

Output:

devicesDiscovered

Type: ObservableCollection<MSUIam>

Collection if devices which responded to the Scan Command.

Return Value:

Type: SocketErrorInfo

Error information, if any while sending the discover command over the network.

Remarks:

Discover Device function does the devices discover operation. It multicasts the Scan command (MSU Who-Is message) over the network. Waits for 3 seconds to collect the response message (MSU I-am message) from the devices. Returns the collection of devices which responded for scan command. (*Refer MSU Message formats definition Chm File in Section 2 of this document for Message Data types*)

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			ObservableCollection<MSUIam> discoveredDevices;
			msuAPI.DiscoverDevices("192.168.2.1”, true, "192.168.2.2”,
			"192.168.2.255”, out discoveredDevices);
		}
	}
}
```

4.2 Connect to Device

Authenticates the user, Server machine for the devices transaction (Upgrade, Downgrade & force upgrade) and returns the response from the device to the corresponding Connect Command.

```
public MSUConnectResponse ConnectToDevice(int authenticationlevel,
                                          string deviceId,
                                          string serverIpAddress,
                                          string hardwareId,
                                          string macId,
                                          string transactionId,
                                          string username,
                                          string password,
                                          FileHeader fileHeader);
```

Namespace:

MSUServer.MSUApiLibrary

Parameters:

authenticationlevel

Type: int

Authentication required for group level or individual level for the device.

serverIPAddress

Type: string

IP address of server required for device authentication

macID

Type: String

Mac ID of the server

Transaction ID

Type: String

Unique ID to be sent to the device. (Each Group has the its own transaction ID.)

username

Type: String

User name of the group or individual device required for authentication.

password

Type: String

Password for the group or individual device required for authentication.

Fileheader

Type: FileHeader

Object of File Header(FileName, HardwareID, ProductID, ModelName, FirmwareVersion, SoftwareVersion, VendorID, PorductName)

Return Value:

Type: MSUConnectResponse

Response from the device for which the Connect Command was sent.

Remarks:

Connect to Device method receives the success or failure status of the authentication from the group of devices or the individual device.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			Dictionary<string,bool> authenticateDevices = msuAPI.ConnectToDevice
			(1,"192.168.2.1”, "84-3A-4B-04-91-C0”, "192.168.2.2”, "Admin”,
			"Password”, fileheader);
		}
	}
}
```

4.3 Disconnect From Device

Disconnects the connected devices for further transaction (Upgrade, Downgrade & force upgrade) and returns the disconnect response from the device.

```
public MSUDisconnectResponse DisconnectDevice(string serverIp,
                                               string transactionId,
                                               string deviceId,
                                               int authenticationlevel);
```

Namespace:

MSUServer.MSUApiLibrary

Parameters:

serverIPAddress

Type: string

IP address of server required for device authentication.

Transaction ID

Type: string

Unique ID, which was created during connection to this device.

`deviceId`

Type: string

IP address of Device which has to be disconnected

`authenticationlevel`

Type: int

Device Authentication Level; (Group Level & Individual Level)

Return Value:

Type: MSUDisconnectResponse

Response from the device about the disconnect command.

Remarks:

Disconnect Operation disconnects the connected device from any transaction with it. This method responds the response from the device to which the Disconnect command was sent. The type “MSUDisconnectResponse” is defined in the MSUMessageFrames Assembly.

Usage Example:

```
using MSUServer.MSUApiLibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
		Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			var responses = msuAPI.DisconnectDevice("192.168.2.11”,
					transactionId, deviceId, authenticationlevel);
		}
	}
}
```

4.4 Get Selected File Properties

Processes the File Selected and returns the File Properties viz. File Checksum, Num of Chunks, Num Of Sequences, File Size, Unpacked Chunks, Unpacked Sequences etc.

```
public FileHandlerError GetSelectedFileProperties(
                                                string filePath,
                                                out FileProperties fileProperties)
```

Namespace:
MSUServer.MSUAPILibrary

Parameters:
filePath
Type: String
Path of the selected File which has to be transmitted to Device.

fileProperties
Type: out FileProperties
Output object of File Properties (*File path, name, size, checksum, Number of chunks, number of sequences, unpacked chunks, unpacked sequences, file stream*)

Return Value:
Type: FileHandlerError
Error occurred while processing the file, if any.

Remarks:
Get Selected File Properties method processes the file passes and returns the file parameters. It checks for whether the file exists in the supplied path, then checks for file accessibility and then starts processing the file for computing the number of chunks, number of sequences and unpacked chunks and sequences. It also computes 32 bit CRC Checksum.

This method outputs the object of FileProperties Type, which contains the file parameters. If any discrepancies found while processing the file, the methods return the discrepancy; else the return value will be none.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
       class CYourClass
       {
              MSUApplicationProtocolInterface msuAPI;
       Ctor() {
                     msuAPI = new MSUApplicationProtocolInterface();
              }

              Method()
              {
                 FileProperties fileProperties;
                 FileHandlerError error = msuAPI.GetSelectedFileProperties
              ("c:\DataFile.ext”, out fileProperties);
               }
       }
}
```

4.5 Notification

Server initiates the MSU process by multicasting notification message. No response is required from the device.

```
public SocketErrorInfo Notify(int transactionID, int hardwareID, FileProperties
                                                  fileProperties, string localIPAddress,
                                                  string multicastAddress,
                                                  int portNum,
                                                  string cmMulticastAddress,
                                                  int cmPortNum,
                                                  int FileNum = 1,
                                                  int transactionType = 1,
                                                  int updateTimeout = 10,
                                                  int seqSizeLimit = 1460,
                                                  int seqNumLimit = 32 );
```

Namespace:

MSUServer.MSUAPILibrary

Parameters:

transactionID

Type: int

Randomly generated number which uniquely represents one MSU cycle (ID created while connecting to the group/device).

fileProperties

Type: FileProperties

Object of File Properties (*File path, name, size, checksum, Number of chunks, number of sequences, unpacked chunks, unpacked sequences, file stream*)

localIPAddress

Type: String

Local IP Address from which the Client-server Transaction should happen.

multicastAddress

Type: String

The client systems that are interested in participating in the MSU process need to join the multicast address.

portNum

Type: int

The client systems that are interested in participating in the MSU process need to listen on this port number to recieve multicast message.

.

cmMulticastAddress

Type: String

The client systems that are interested in participating in the CM either in SCM or CCM need to join the multicast address.

cmPortNum

Type: int

The client systems that are interested in participating in the CM either in SCM or CCM need to listen on this port to recieve CM multicast address

FileNum

Type: int

Number of file ‘s under transfer. Default value is 1.

transactionType

Type: int

Transaction type for upgrade, downgrade or forceupgrade. Default value is 1.

updateTimeout

Type: int

On communication failure for a period greater than this timeout value, the client must exit from the current process and become available for next update cycle without participating for the rest of the current update process.

seqSizeLimit

Type: int

Maximum number of sequences in the first chunk upto the last but one chunk.

seqNumberLimit

Type: int

Maximum size in bytes of the data payload of a single packet sequence.

Return Value:

Type: SocketErrorInfo

Error information, if any while sending the Notification over the network.

Remarks:
MSU Server sends the notification message before initiating the file transfer process. The notification message is sent as a multicast message and no response is sent from the clients to the server.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			Helper.Instance.msuAPI.Notify(nTransactionId,
			nhardwaretype, _selectedPackage.FilePropertiesCollection[0],
			 "192.168.2.11”, "239.254.1.5”,
			 568,"239.254.1.5”, 0);
		}
	}
}
```

4.6 File Transfer Operation

This method transfers the served file to the devices which are connected.

```
public bool FileTransferToDevicesGroup(UInt16 nFileNumber,
                                       FileProperties fileProperties,
                                       MSUConfigParameters configParameters,
                                       string sRemoteIp, int nRemotePort,
                                       string sLocalIp, int nLocalPort,
                                       int nHardwareId, int maxNumOfSequences = 32);
```

Namespace:
MSUServer.MSUAPILibrary

Parameters:
nFileNumber
Type: int
If there are multiple files to be transferred to the client, specify the file number for which this method was invoked.

fileProperties
Type: FileProperties

File Properties (*File path, name, size, checksum, Number of chunks, number of sequences, unpacked chunks, unpacked sequences, file stream*) for the file which has to be transferred.

configParameters

Type: MSUConfigParameters

MSU Protocol configuration parameters reference (SCM, CCM, timeout, sequence Limit, SCM Retry, CCM Retry…).

sRemoteIp

Type: string

Multicast address on which the file has to be transferred.

nRemotePort

Type: int

Port number on which the file has to be transferred.

sLocalIp

Type: string

Local IP Address from which the Client-server Transaction should happen.

nLocalPort

Type: int

Port number from which the file has to be transferred.

nHardwareId

Type: int

Hardware Type Identifier (eg. 0 – M580, 1 – M168, 2 – HMI etc.)

maxNumOfSequences

Type: int

Sequence Limit configured for the file transaction.

Return Value:

Type: bool

File transfer send status. False, if any Error occurred while processing the file.

Remarks:

This method starts sending the data packet from the first chunk to the end of the chunk in the served file and its properties and returns if there were any error while sending the file to the selected group of devices.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			msuAPI.FileTransferToDevicesGroup(nFileNumber, fileProperties,
			configParameters, "239.254.1.6”, 569, "192.168.2.11”, 569,
			(int)hwType, 32);
		 }
	}
}
```

4.7 Get File Transfer Progress Parameters

This method updates the caller with the file transfer progress parameters. It gives the file number being transferred, chunk number in the file being transferred, sequence number in the chunk being transferred and whether the transfer state is complain mode.

```
public void GetDataTransferProgressParameters(int nHardwareId,
                                    out int nFileNum, out int nChunkNum,
                                    out int nSeqNum, out bool bComplainModeInProgress);
```

Namespace:

MSUServer.MSUAPILibrary

Parameters:

nHardwareId

Type: int

Hardware type identifier for which the transaction progress parameters has to be updated.

Ouput:

nFileNum

Type: out int

File number being transferred.

nChunkNum

Type: out int

Chunk number in the file being transferred.

nSeqNum

Type: out int

Sequence number in the chunk being transferred.

bComplainModeInProgress

Type: out int

Flag indicates whether the transfer state is complaining mode.

Return Value:

None.

Remarks:

This method shall be used to update the progress bar and progress parameters in the UI for the file transfer command. This method updates the caller with the file transfer progress parameters. It gives the file number being transferred, chunk number in the file being transferred, sequence number in the chunk being transferred and whether the transfer state is complain mode.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			int nChunkNo, nSeqNo, nFileNo;
			bool bComplainModeInProgress;
			msuAPI.GetDataTransferProgressParameters(hwType, out nFileNo, out
		nChunkNo, out nSeqNo, out bComplainModeInProgress);

			// TODO: Update UI here with the returned parameters

		 }
	}
}
```

4.8 Abort File Transfer Operation

This method aborts the file transfer Operation which was initiated for a group of devices. This method shall require file number and the hardware type ID (group ID) which has to be aborted.

```
public void AbortFileTransferOperation(int nHardwareId, UInt16 nFileNumber);
```

Namespace:
MSUServer.MSUAPILibrary

Parameters:
nHardwareId
Type: int
Hardware type identifier for which the transaction progress parameters has to be updated.

nFileNumber
Type: int
If there are multiple files being transferred to the device and abort initiated, then the caller should specify for which file the abort command is sent.

Return Value:
None.

Remarks:
This method aborts the file transfer Operation which was initiated for a group of devices. This method shall require file number and the hardware type ID (group ID) which has to be aborted. Abort command operated for each file in the package which is being transferred.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			msuAPI.AbortFileTransferOperation((int)hwType, fileNumber);
		 }
	}
}
```

4.9 Request Transaction Status

After the file transfer operation server shall request for the status of the transaction to the devices over the default network and returns the collection of devices transaction status responses.

```
public ObservableCollection<MSUTransmitStatusResponse> RequestTrasactionStatus(
                    string sRemoteIp, int nRemotePort, string sLocalIp, int nLocalPort,
                    int nTransactionId);
```

Namespace:

MSUServer.MSUAPILibrary

Parameters:

filePath

Type: String

Path of the selected File which has to be transmitted to Device.

sRemoteIp

Type: string

Multicast address on which the request has to be transferred.

nRemotePort

Type: int

Port number on which the request has to be transferred.

sLocalIp

Type: string

Local IP Address from which the request has to be sent.

nLocalPort

Type: int

Port number from which the request has to be sent.

nTransactionId

Type: int

Randomly generated number which uniquely represents one MSU cycle (ID created while connecting to the group/device).

Return Value:

Type: ObservableCollection<MSUTransmitStatusResponse>

The collection of devices transaction status responses.

Remarks:

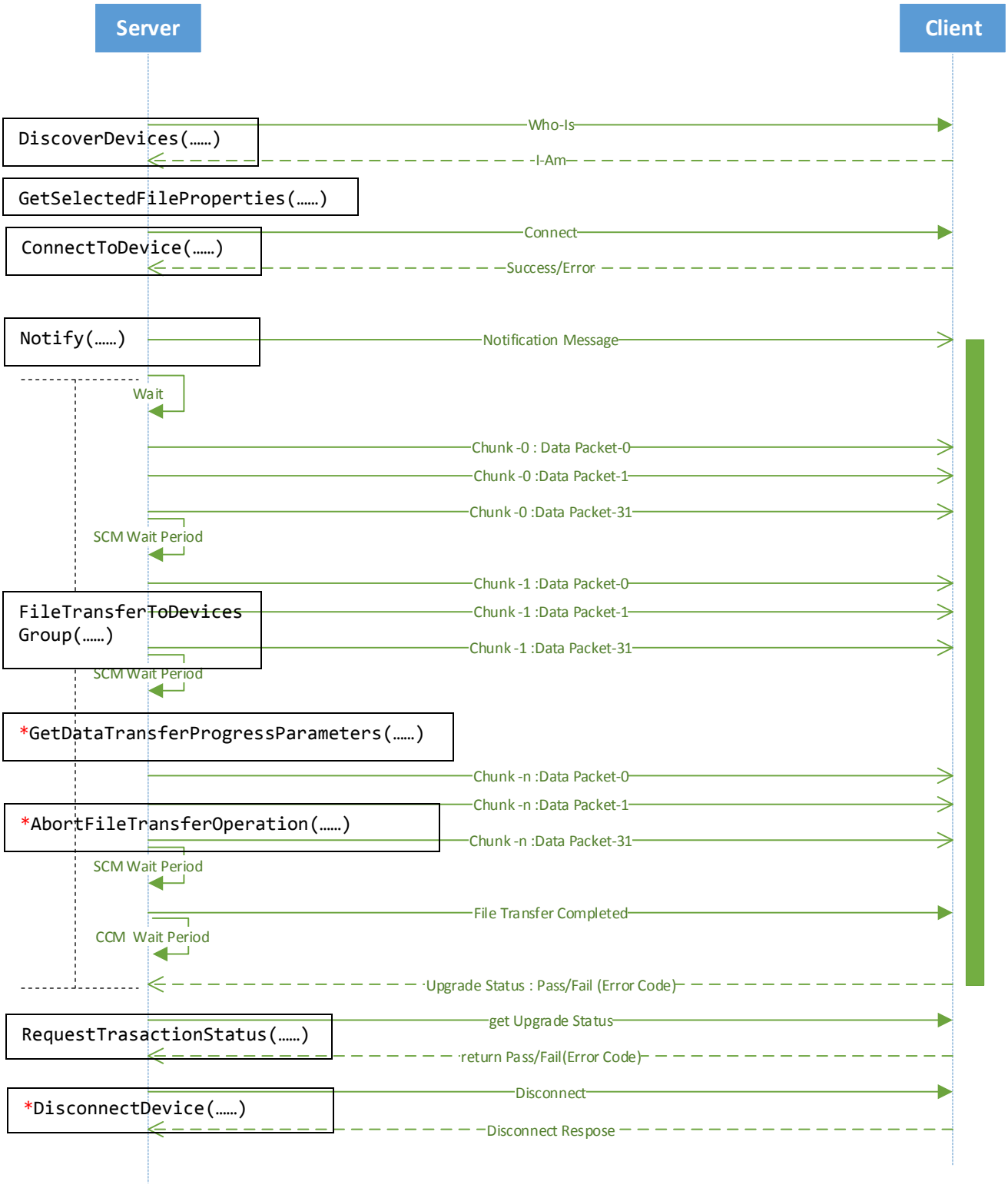
After the file transfer operation server shall request for the status of the transaction to the devices over the default network and returns the collection of devices transaction status responses.

Usage Example:

```
using MSUServer.MSUAPILibrary;
using MSUServer.MSUMessageFrames;
namespace YourNamespace
{
	class CYourClass
	{
		MSUApplicationProtocolInterface msuAPI;
	Ctor() {
			msuAPI = new MSUApplicationProtocolInterface();
		}

		Method()
		{
			msuAPI.RequestTrasactionStatus("239.254.1.2”, 567,
			"192.168.2.11”, 567, nTransactionId);
		 }
	}
}
```

5 SEQUENCE OF CALLING APIS



* Optional (*to be called on a separate Thread*)

6 NOTES

6.1 Glossary

To be updated

6.2 Acronyms and Abbreviations

To be updated