New SkyeTek Protocol V3 Messages

June 10, 2006

SkyeTek, Inc.
11030 Circle Point Road, Suite 300
Westminster, CO 80020

720-565-0441 phone
720-565-8989 fax

Sales
sales@skyetek.com

Technical Support
techsupport@skyetek.com

Table of Contents

1 Introduction

The SkyeTek Protocol V3 uses new messages not used in V2 of the protocol. Not all of these messages have been defined nor implemented. This document defines the new messages and their parameters as implemented in the C API with the expectation that either the firmware will be implement the messages as defined in this document or this document will be updated to reflect the implementation, when complete.

2 Reference

"SkyeTek Protocol v3 Specification," SkyeTek Engineering, 4/20/2006.

3 Reader Functions

These are messages used to get and set information about the reader.

3.1 StoreKey

This function stores a key on the reader at a specific address. The key is used in various security-related functions, such as authentication. The application passes in the address of the memory location in which to store the key, a buffer containing the key, and the size in bytes of the key.

```
Command:      0x0601 (Store Key)
Flags:        0x0020 (CRC_F)
Address:      0xXXXX (Address supplied by application)
NumBlocks:    0x0001 (One Block)
DataLength:   0xXXXX (Length of the key)
Data:         <data> (Up to 1024 bytes in the key)
```

The reader responds with:

```
Response:     0x0601 (Store Key Pass) or 0x81C1 (Select Tag Loop Off)
```

3.2 LoadKey

Once a key is stored on the reader, the application can call it to be loaded into memory to be used by the reader in the security related functions. The application passes in the address of the memory location in which the key is stored and loaded from.

```
Command:      0x0602 (Load Key)
Flags:        0x0020 (CRC_F)
Address:      0xXXXX (Address supplied by application)
NumBlocks:    0x0001 (One Block)
```

The reader responds with:

```
Response:     0x0602 (Load Key Pass) or 0x81C1 (Select Tag Loop Off)
```

.

3.3 LoadDefaults

This function resets the system parameters on the reader by loading in the default settings.

```
Command:      0x1101 (Load Defaults)
Flags:        0x0020 (CRC_F)
```

The reader responds with:

```
Response:     0x1101 (Load Defaults Pass) or 0x81C1 (Select Tag Loop Off)
```

3.4 ResetDevice

This function resets the reader.

```
Command:      0x1102 (Reset Device)
Flags:        0x0020 (CRC_F)
```

The reader responds with:

```
Response:     0x9102 (Cannot Reset Device) or 0x81C1 (Select Tag Loop Off)
```

If the reader resets the device, it does not send a response.

3.5 Bootload

This function causes the reader to bootload.

```
Command:      0x1103 (Bootload)
Flags:        0x0020 (CRC_F)
```

The reader responds with:

```
Response:     0x1103 (Enter Bootload), 0x9103 (Bootload Fail), or 0x81C1 (Select Tag Loop Off)
```

3.6 GetSystemParameter

This gets the current value of the given system parameter. The parameter retrieved is specified in the address block using one of the system parameters defined in Appendix A.

```
Command:      0x1201 (Read System Parameter)
Flags:        0x0020 (CRC F)
Address:      0xXXXX (Parameter to Read – see Appendix A)
NumBlocks:    0x0001 (One Block)
```

The reader responds with:

```
Response:     0x1201 (Read System Parameter Pass), 0x9201 (Read System Parameter Fail),
              or 0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the parameter value)
Data:         <data> (Up to 1024 bytes of data)
```

3.7 SetSystemParameter

This sets the value of the given system parameter. The parameter set is specified in the address block using one of the system parameters defined in Appendix A. The data buffer contains the value to set and the size parameter indicates the size in bytes of the buffer.

```
Command:      0x1202 (Write System Parameter)
Flags:        0x0020 (CRC F)
Address:      0xXXXX (Parameter to Write – see Appendix A)
NumBlocks:    0x0001 (One Block)
DataLength:   0xXXXX (Length of the parameter value)
Data:         <data> (Up to 1024 bytes of data)
```

The reader responds with:

```
Response:     0x1202 (Write System Parameter Pass),
              0x9202 (Write System Parameter Fail), or
              0x81C1 (Select Tag Loop Off)
```

3.8 GetDefaultSystemParameter

This gets the default value of the given system parameter. This is the value that the reader will use when Load Defaults is called. The parameter retrieved is specified in the address block using one of the system parameters defined in Appendix A.

```
Command:      0x1302 (Read Default System Parameter)
Flags:        0x0020 (CRC F)
Address:      0xXXXX (Parameter to Read – see Appendix A)
NumBlocks:    0x0001 (One Block)
```

The reader responds with:

```
Response:     0x1302 (Read Default System Parameter Pass),
              0x9302 (Read Default System Parameter Fail),
              or 0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the parameter value)
Data:         <data> (Up to 1024 bytes of data)
```

3.9 SetDefaultSystemParameter

This sets the value of the default system parameter. The new value will be what the reader will use when Load Defaults is called. The parameter set is specified in the address block using one of the system parameters defined in Appendix A. The data buffer contains the value to set and the size parameter indicates the size in bytes of the buffer.

```
Command:      0x1301 (Write Default System Parameter)
Flags:        0x0020 (CRC_F)
Address:      0xXXXX (Parameter to Write – see Appendix A)
NumBlocks:    0x0001 (One Block)
DataLength:   0xXXXX (Length of the parameter value)
Data:         <data> (Up to 1024 bytes of data)
```

The reader responds with:

```
Response:     0x1301 (Write Default System Parameter Pass),
              0x9301 (Write Default System Parameter Fail), or
              0x81C1 (Select Tag Loop Off)
```

3.10 AuthenticateReader

This authenticates the reader using the credentials passed in. The data contains the credentials and the data length indicates how many bytes are in the data buffer.

```
Command:      0x1401 (Authenticate Reader)
Flags:        0x0020 (CRC_F)
DataLength:   0xXXXX (Length of the credentials)
Data:         <data> (Up to 1024 bytes of data)
```

The reader responds with:

```
Response:     0x1401 (Authenticate Reader Pass),
              0x9401 (Authenticate Reader Fail), or
              0x81C1 (Select Tag Loop Off)
```

3.11 EnableDebug

This function enables debugging on the reader. Debug output will be written to the debug terminal port on the reader.

```
Command:      0x1402 (Enable Debug)
Flags:        0x0020 (CRC_F)
```

The reader responds with:

```
Response:     0x1402 (Enable Debug Pass),
              0x9402 (Enable Debug Fail), or
              0x81C1 (Select Tag Loop Off)
```

3.12 DisableDebug

This function disables debugging on the reader. Debug output will no longer be written to the debug terminal port on the reader.

```
Command:      0x1403 (Disable Debug)
Flags:        0x0020 (CRC_F)
```

The reader responds with:

```
Response:     0x1403 (Disable Debug Pass),
              0x9403 (Disable Debug Fail), or
              0x81C1 (Select Tag Loop Off)
```

4 Tag Functions

The first set of the following tag functions may be used to discover tags in the field range of the reader. Once a tag is discovered, the subsequent functions may be performed on the tag itself, passing in the tag ID.

4.1 SelectTag

This function selects a tag in the field. To The application specifies the tag type of the tag to select and/or the tag ID itself. If no tag ID is specified, the reader will select the first tag of the type given. If the tag type is set to one of the auto-detect tag types, the tag type is returned. The tag ID is returned if the call is successful.

```
Command:      0x0101 (Select Tag)
Flags:        0x0020 (CRC_F) and optional flags: 0x0001 (LOOP_F), 0x0002 (INV_F), and 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0101 (Select Tag Pass),
              0x8101 (Select Tag Fail),
              0x01C1 (Select Tag Loop On), or
              0x81C1 (Select Tag Loop Off)
TagType:      0xXXXX (If the request TagType was an auto-detect type)
TagLength:    0xXXXX (Length of Tag ID)
Tag:          <Up to 16 bytes> (Tag ID)
```

4.2 ReadTagData

This reads the specified number of blocks of data from a tag. The tag must be in a selected state, by calling Select Tag first.

```
Command:      0x0102 (Read Tag Data)
Flags:        0x0020 (CRC F) and optional 0x0040 (TID F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID F is set)
Address:      0xXXXX (Starting Address to Read)
NumBlocks:    0xXXXX (Number of Blocks to Read)
```

The reader responds with:

```
Response:     0x0102 (Read Tag Data Pass),
              0x8102 (Read Tag Data Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

4.3 WriteTagData

This writes data to the tag. It starts writing at the address specified by and writes over up to the number of blocks given. If the LOCK_F flag is set, it locks the blocks.

```
Command:      0x0103 (Write Tag Data)
Flags:        0x0020 (CRC_F) and optional tags: 0x0040 (TID_F) and 0x0004 (LOCK_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
Address:      0xXXXX (Starting Address to Write)
NumBlocks:    0xXXXX (Number of Blocks to Write)
DataLength:   0xXXXX (Length of the data unless LOCK F is set)
Data:         <data> (Up to 1024 bytes of data unless LOCK_F is set)
```

The reader responds with:

```
Response:     0x0103 (Write Tag Data Pass),
              0x8103 (Write Tag Data Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.4 ActivateTagType

This function activates the tags of the given tag type.

```
Command:      0x0104 (Activate Tag Type)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0104 (Activate Tag Type Pass),
              0x8104 (Activate Tag Type Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.5 DeactivateTagType

This deactivates the tags of the given tag type.

```
Command:      0x0105 (Deactivate Tag Type)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0105 (Deactivate Tag Type Pass),
              0x8105 (Deactivate Tag Type Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.6 SetTagToReaderDataRate

This sets the tag data rate to the reader data rate given. The application passes in the data rate in the data field.

```
Command:      0x0106 (Set Tag To Reader Data Rate)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

The reader responds with:

```
Response:     0x0106 (Set Tag To Reader Data Rate Pass),
              0x8106 (Set Tag To Reader Data Rate Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.7 SetReaderToTagDataRate

This sets the reader data rate to the tag data rate given. The application passes in the data rate in the data field.

```
Command:      0x010D (Set Reader To Tag Data Rate)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

The reader responds with:

```
Response:     0x010D (Set Reader To Tag Data Rate Pass),
              0x810D (Set Reader To Tag Data Rate Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.8 GetTagInfo

This function gets the tag information for the specified tag. It returns the tag handle and the tag ID.

```
Command:      0x0107 (Get Tag Info)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0107 (Get Tag Info Pass),
              0x8107 (Get Tag Info Fail),
              0x81C1 (Select Tag Loop Off)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of Tag ID)
Tag:          <Up to 16 bytes> (Tag ID)
```

4.9 GetLockStatus

This checks the lock state on the given address to determine whether or not it is locked. The application passes in the tag handle and the address to check The reader returns one byte set to 0x01 if the address is locked or 0x00 if not.

```
Command:      0x0108 (Get Lock Status)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
Address:      0xXXXX (Starting Address to Read)
```

The reader responds with:

```
Response:     0x0108 (Get Lock Status Pass),
              0x8108 (Get Lock Status Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0x0001 (Only One Byte)
Data:         0x0X   (One Byte)
```

4.10 KillTag

This call issues a “kill” command to the given. Once the kill command is given, the tag is no longer selectable.

```
Command:      0x0109 (Kill Tag)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0109 (Kill Tag Pass),
              0x8108 (Kill Tag Fail),
              0x81C1 (Select Tag Loop Off)
```

4.11 ReviveTag

This revives a tag that his been "killed" so that it is selectable again. The application passes in a handle to the tag to revive.

```
Command:      0x010A (Revive Tag)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x010A (Revive Tag Pass),
              0x810A (Revive Tag Fail),
              0x81C1 (Select Tag Loop Off)
```

4.12 EraseTag

This erases all blocks of memory on the given tag.

```
Command:      0x010B (Erase Tag)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x010B (Erase Tag Pass),
              0x810B (Erase Tag Fail),
              0x81C1 (Select Tag Loop Off)
```

4.13 FormatTag

This formats the tag for writing. Some tags require the memory to be formatted before it is written to.

```
Command:      0x010C (Format Tag)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
```

```
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x010C (Format Tag Pass),
              0x810C (Format Tag Fail),
              0x81C1 (Select Tag Loop Off)
```

4.14 AuthenticateTag

This authenticates the tag using the key or password set by a previous command.

```
Command:      0x0201 (Authenticate Tag)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0201 (Authenticate Tag Pass),
              0x8201 (Authenticate Tag Fail),
              0x81C1 (Select Tag Loop Off)
```

4.15 SendTagPassword

This causes the tag password to be sent to the tag. The password must be loaded by a previous command.

```
Command:      0x0201 (Send Tag Password)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0201 (Send Tag Password Pass),
              0x8201 (Send Tag Password Fail),
              0x81C1 (Select Tag Loop Off)
```

4.16 GetApplicationIDs

This gets the all application IDs available on the given tag. The application sends the request and the reader and the reader enters a loop, responding with application IDs until there are no more. When done, the reader sends the fail response to indicate no more IDs.

```
Command:      0x0301 (Get Application Ids)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
```

```
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0301 (Get Application Ids Pass) or 0x8301 (Get Application Ids Fail)
DataLength:   0xXXXX (Length of ID)
Data:         <bytes> (The ID)
```

4.17 SelectApplication

This selects the application on the tag with the given application ID.

```
Command:      0x0302 (Select Application)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the Application ID)
Data:         <data> (Application ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0301 (Select Application Pass),
              0x8301 (Select Application Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.18 CreateApplication

This creates a new application with the given application ID on the given tag.

```
Command:      0x0303 (Create Application)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the Application ID)
Data:         <data> (Application ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0303 (Create Application Pass),
              0x8303 (Create Application Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.19 GetFileIDs

This gets the all file IDs available on the given tag. The application sends the request and the reader and the reader enters a loop, responding with file IDs until there are no more. When done, the reader sends the fail response to indicate no more IDs.

```
Command:      0x0401 (Get File Ids)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0401 (Get File Ids Pass) or 0x8401 (Get File Ids Fail)
DataLength:   0xXXXX (Length of ID)
Data:         <bytes> (The ID)
```

4.20 SelectFile

This selects the given file on the given tag.

```
Command:      0x0402 (Select File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0402 (Select File Pass),
              0x8402 (Select File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.21 CreateFile

This creates a new file with the given file ID on the given tag.

```
Command:      0x0403 (Create File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0403 (Create File Pass),
              0x8403 (Create File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.22 GetFileSettings

This function queries the tag for the file settings of file identified by the given file ID.

```
Command:      0x0404 (Get File Settings)
Flags:        0x0020 (CRC F) and optional 0x0040 (TID F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0404 (Get File Settings Pass),
              0x8404 (Get File Settings Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

4.23 ChangeFileSettings

This sets the file settings for the given file on the given tag to those passed in. The first part of the data buffer is the file ID and the second part is the file settings.

```
Command:      0x0405 (Change File Settings)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
DataLength:   0xXXXX (Length of the File settings)
Data:         <data> (Data to write up to 1024 bytes minus file ID length)
```

The reader responds with:

```
Response:     0x0405 (Change File Settings Pass),
              0x8405 (Change File Settings Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.24 ReadFile

This reads the given file on the given tag and returns a buffer with the file's data.

```
Command:      0x0406 (Read File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0406 (Read File Pass),
              0x8406 (Read File Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

4.25 WriteFile

This writes the file to the given tag. The application passes in the handle to the tag, the ID of the file to write to, the data to write and the size of the data to write. The first part of the data buffer is filled with the file ID and the second part is filled with the file settings.

```
Command:      0x0407 (Write File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
DataLength:   0xXXXX (Length of the data to write)
Data:         <data> (Data to write up to 1024 bytes minus file ID length)
```

The reader responds with:

```
Response:     0x0407 (Write File Pass),
              0x8407 (Write File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.26 DeleteFile

This deletes the given file from the given tag. Files that are deleted may no longer be written to.

```
Command:      0x0408 (Delete File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
```

```
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0408 (Delete File Pass),
              0x8408 (Delete File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.27 ClearFile

This clears the contents of the file but does not remove the file from the tag. Files that are cleared may be written to.

```
Command:      0x0409 (Clear File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID)
Data:         <data> (File ID up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0409 (Clear File Pass),
              0x8409 (Clear File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.28 IncrementValueFile

This function increments the value in the given file on the given tag by the given amount. The first part of the data buffer is the file ID and the second part is the value to increment by.

```
Command:      0x040A (Increment Value File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID plus 2 bytes for value)
Data:         <data> (File ID up to 1022 bytes)
Value:        0xXXXX (Value to increment in two bytes)
```

The reader responds with:

```
Response:     0x040A (Increment Value File Pass),
              0x840A (Increment Value File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.29 DecrementValueFile

This function decrements the value in the given file on the given tag by the given amount. The first part of the data buffer is the file ID and the second part is the value to decrement by.

```
Command:      0x040B (Decrement Value File)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the File ID plus 2 bytes for value)
Data:         <data> (File ID up to 1022 bytes)
Value:        0xXXXX (Value to decrement in two bytes)
```

The reader responds with:

```
Response:     0x040B (Decrement Value File Pass),
              0x840B (Decrement Value File Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.30 EnableEAS

This enables Electronic Article Surveillance (EAS) on the given tag.

```
Command:      0x0501 (Enable EAS)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0501 (Enable EAS Pass),
              0x8501 (Enable EAS Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.31 DisableEAS

This disables Electronic Article Surveillance (EAS) on the given tag.

```
Command:      0x0502 (Disable EAS)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
```

```
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0502 (Disable EAS Pass),
              0x8502 (Disable EAS Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.32 ScanEAS

This scans the given tag for its current Electronic Article Surveillance (EAS) state. The reader returns one byte set to 0x01 if EAS is enabled or 0x00 if not.

```
Command:      0x0503 (Scan EAS)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0503 (Scan EAS Pass),
              0x8503 (Scan EAS Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0x0001 (Only One Byte)
Data:         0x0X   (One Byte: 0x01 = Enabled, 0x00 = Disabled)
```

4.33 ReadAFI

This reads the value of the AFI and returns it in a buffer.

```
Command:      0x0504 (Read AFI)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0504 (Read AFI Pass),
              0x8504 (Read AFI Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

4.34 WriteAFI

This writes the given AFI value to the given tag. The application passes in the AFI data to write and the size of that data.

```
Command:      0x0505 (Write AFI)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the AFI data)
Data:         <data> (AFI data up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0505 (Write AFI Pass),
              0x8505 (Write AFI Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.35 ReadDSFID

This reads the value of the DSFID and returns it in a buffer.

```
Command:      0x0506 (Read DSFID)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
```

The reader responds with:

```
Response:     0x0506 (Read DSFID Pass),
              0x8506 (Read DSFID Fail), or
              0x81C1 (Select Tag Loop Off)
DataLength:   0xXXXX (Length of the data)
Data:         <data> (Up to 1024 bytes of data)
```

4.36 WriteDSFID

This writes the given DSFID value to the given tag. The application passes in the DSFID data to write and the size of that data

```
Command:      0x0507 (Write DSFID)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0xXXXX (Length of the DSFID data)
Data:         <data> (DSFID data up to 1024 bytes)
```

The reader responds with:

```
Response:     0x0507 (Write DSFID Pass),
              0x8507 (Write DSFID Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.37 CreditValue

This function credits the given address block on the given tag with the given value.

```
Command:      0x0508 (Credit Value)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0x0002 (Value is two bytes)
Data:         0xXXXX (Value in two bytes)
```

The reader responds with:

```
Response:     0x0508 (Credit Value Pass),
              0x8508 (Credit Value Fail), or
              0x81C1 (Select Tag Loop Off)
```

4.38 DebitValue

This debits the given address block on the given tag with the given value.

```
Command:      0x0508 (Debit Value)
Flags:        0x0020 (CRC_F) and optional 0x0040 (TID_F)
TagType:      0xXXXX (One of the tag types in Appendix B)
TagLength:    0xXXXX (Length of tag ID if TID_F is set)
TagID:        <Up to 16 bytes> (Tag ID if the TID_F is set)
DataLength:   0x0002 (Value is two bytes)
Data:         0xXXXX (Value in two bytes)
```

The reader responds with:

```
Response:     0x0508 (Debit Value Pass),
              0x8508 (Debit Value Fail), or
              0x81C1 (Select Tag Loop Off)
```

APPENDIX A: System Parameters

The system parameters is used when reading or setting system parameters.

Hex Value	Description
0x0000	The serial number of the reader.
0x0001	The firmware version number.
0x0002	The hardware version number.
0x0003	The product code of the reader.
0x0004	The reader ID value.
0x0005	The reader name.
0x0006	The host interface type.
0x0007	The host interface baud rate.
0x0008	User port direction.
0x0009	User port value.
0x000A	Multiplexer control.
0x000C	Operating mode.
0x000D	Encryption Scheme.
0x000E	HMAC Scheme.
0x000F	Peripheral Select.
0x0010	Tag population optimization.
0x0011	System command retry.
0x0012	Tx power control.
0x0014	Test parameter.
0x0015	Test parameter.

APPENDIX B: Tag Types

The following tag types are used when reading or writing tags based on type.

Hex Value	Description
0x0000	AUTO_DETECT
0x0001	ISO15693
0x0002	I_CODE1
0x0003	TAG_IT_HF
0x0004	ISO14443A
0x0005	ISO14443B
0x0006	PICOTAG
0x0008	GEMWAVE
0x0009	LTO_CM
0x000A	MIFARE_ULTRALIGHT
0x000B	LRI64
0x0081	CLASS1
0x0082	CLASS0
0x0083	CLASS0PIMP
0x0084	CLASS0PMAT
0x0085	ISO180006A
0x0086	ISO180006B
0x0087	CLASS1GEN2
0x0088	EM4222
0x0089	EM4022
0x008A	UCODE
0x008B	TAGIDU
0x008C	SCS
0x0100	ISO_15693_AUTO_DETECT
0x0110	TI_15693_AUTO_DETECT
0x0111	TAGIT_HF1_STANDARD
0x0112	TAGIT_HF1_PRO
0x0113	TAGIT_HF1_PLUS
0x0120	PHILIPS_15693_AUTO_DETECT
0x0121	ICODE_SLI_SL2
0x0130	ST_15693_AUTO_DETECT
0x0131	ISO_LRI64
0x0132	LRI512
0x0140	EM_15693_AUTO_DETECT
0x0141	EM4006
0x0142	EM4034
0x0143	EM4035_CRYPTO
0x0144	EM4135
0x0150	INFINEON_15693_AUTO_DETECT
0x0151	MYD2K
0x0152	MYD2KS
0x0153	MYD10K
0x0154	MYD10KS
0x0200	ISO_14443A_AUTO_DETECT
0x0210	PHILIPS_14443A_AUTO_DETECT
0x0211	ISO_MIFARE_ULTRALIGHT
0x0212	MIFARE_1K
0x0213	MIFARE_4K
0x0214	MIFARE_DESFIRE
0x0215	MIFARE_PROX
0x0220	INNOVISION_14443A_AUTO_DETECT
0x0221	JEWEL
0x0300	ISO_14443B_AUTO_DETECT
0x0310	ATMEL_14443B_AUTO_DETECT
0x0311	CRYPTORF_1K
0x0312	CRYPTORF_2K
0x0313	CRYPTORF_4K
0x0314	CRYPTORF_8K

0x0315	CRYPTORF_16K
0x0316	CRYPTORF_32K
0x0317	CRYPTORF_64K
0x0318	AT88RF001
0x0319	AT88RF020
0x0330	SAMSUNG_14443B_AUTO_DETECT
0x0331	S3C89K8
0x0332	S3C89V5
0x0333	S3C89V8
0x0334	S3CC9G4
0x0335	S3CC9GC
0x0336	S3CC9GW
0x0337	S3CC9W4
0x0338	S3CC9W9
0x0400	ISO_18000_3_MODE1_AUTO_DETECT
0x0410	M1_TI_15693_AUTO_DETECT
0x0411	M1_TAGIT_HF1_STANDARD
0x0412	M1_TAGIT_HF1_PRO
0x0413	M1_TAGIT_HF1_PLUS
0x0420	M1_PHILIPS_15693_AUTO_DETECT
0x0421	M1_ICODE_SLI_SL2
0x0430	M1_ST_15693_AUTO_DETECT
0x0431	M1_LRI64
0x0432	M1_LRI512
0x0440	M1_EM_15693_AUTO_DETECT
0x0441	M1_EM4006
0x0442	M1_EM4034
0x0443	M1_EM4035_CRYPTO
0x0444	M1_EM4135
0x0450	M1_INFINEON_15693_AUTO_DETECT
0x0451	M1_MYD2K
0x0452	M1_MYD2KS
0x0453	M1_MYD10K
0x0454	M1_MYD10KS
0x0500	ISO_18000_3_MODE1_EXTENDED_AUTO_DETECT
0x0510	RFU
0x0511	TAGSYS
0x0600	ISO_18000_3_MODE2_AUTO_DETECT
0x0610	INFINEON_AUTO_DETECT
0x0611	INFINEON_PJM_TAG
0x0700	ISO_18092_AUTO_DETECT
0x0800	ISO_21481_AUTO_DETECT
0x0900	HF_PROPRIETARY_RFU
0x0901	TAGIT_HF
0x0902	ICODE1
0x0903	HF_EPC
0x0904	LTO_PHILIPS
0x0905	LTO_ATMEL
0x0906	FELICIA
0x0907	PICOTAG_2K
0x0908	PICOTAG_16K
0x0909	PICOTAG_2KS
0x0910	PICOTAG_16KS
0x0911	HID_ICLASS
0x0912	GEMWARE_C210
0x0913	GEMWARE_C220
0x0914	GEMWARE_C240
0x0915	SR176
0x8000	EPC_CLASS0_AUTO_DETECT
0x8010	SYMBOL_CLASS0_AUTO_DETECT
0x8011	MATRICS_CLASS0
0x8012	MATRICS_CLASS0_PLUS
0x8020	IMPINJ_CLASS0_AUTO_DETECT
0x8021	ZUMA
0x8100	EPC_CLASS1_GEN1_AUTO_DETECT
0x8110	ALIEN_C1G1_AUTO_DETECT

0x8111	QUARK
0x8112	OMEGA
0x8113	LEPTON
0x8120	ST_MICRO_C1G1_AUTO_DETECT
0x8121	XRA00
0x8200	ISO_18000_6C_AUTO_DETECT
0x8210	IMPINJ_C1G2_AUTO_DETECT
0x8211	MONZA
0x8220	PHILIPS_C1G2_AUTO_DETECT
0x8221	UCODE_EPC_G2
0x8230	ST_C1G2_AUTO_DETECT
0x8231	XRAG2
0x8300	ISO_18000_6B_AUTO_DETECT
0x8310	PHILIPS_18000_6B_AUTO_DETECT
0x8311	UCODE_1_19
0x8312	UCODE_HSL
0x8400	ISO_18000_6A_AUTO_DETECT
0x8401	EM_6A
0x08500	UHF_PROPRIETARY_RFU
0x8501	NEOLOGY
0x8502	PROP_EM4222
0x8503	PROP_EM4223