# *Inspection and Sanitization Guidance for National Imagery Transmission Format (NITF)*

Version 1.1

3 March 2016

**National Security Agency**
**Information Assurance Capabilities**
**9800 Savage Rd, Suite 6699**
**Ft. George G. Meade. MD 20755**

**Authored/Released by:**
**Unified Cross Domain Capabilities Office**
**cds_tech@nsa.gov**

# DOCUMENT REVISION HISTORY

| Date | Version | Description |
|---|---|---|
| 6/20/2013 | 1.0 | Final |
| 3/3/2016 | 1.1 | Changes after review in 2016. |
| 12/13/2017 | 1.1 | Updated Contact information, IAC Logo, Cited Trademarks and Copyrights, Expanded Acronyms, and added Legal Disclaimer |

**DISCLAIMER OF WARRANTIES AND ENDORSEMENT**

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer or otherwise, does not constitute or imply its endorsement, recommendation or favoring by the United States Government and this guidance shall not be used for advertising or product endorsement purposes.

# EXECUTIVE SUMMARY

This *Inspection and Sanitization Guidance (ISG) for National Imagery Transmission Format (NITF)* document provides guidelines and specifications for developing file inspection and sanitization software for NITF files, which are formally defined by the National Imagery Transmission Format Standard (NITFS). The latest NITF 2.1 standard is defined in MIL-STD-2500C. NITF files contain numerous segments of data that include images, graphics, text, as well as custom data in a strict format. As with prior ISG documents, this document is concerned with data hiding, data disclosure, and data attack risks. For example, although the NITF standard is well-defined and contains detail down to the byte level, there remain fields that include metadata, conditional information, and variable length content that require inspection to ensure nothing is hidden within the file. The nature of NITF files is to include a variety of imagery and associated data that could potentially be displayed to an end user. Information can be selectively displayed to the user based on capability and the information that was requested. With potentially a large amount of data located in these files, inspection and sanitization is key to ensuring that information contained in the file is authorized for display to the user and that the data cannot be used to attack the system.

# TABLE OF CONTENTS


**1. SCOPE ..... 1-1**
1.1 PURPOSE OF THIS DOCUMENT ..... 1-1
1.2 INTRODUCTION ..... 1-1
1.3 BACKGROUND ..... 1-1
1.4 DOCUMENT ORGANIZATION ..... 1-2
1.5 RECOMMENDATIONS ..... 1-2
1.5.1 Actions ..... 1-2
1.5.2 Action Options ..... 1-4
1.5.3 Naming Convention for Recommendations ..... 1-5
1.6 DATA TRANSFER GUIDANCE ..... 1-5
1.7 DOCUMENT LIMITATIONS ..... 1-5
1.7.1 Covert Channel Analysis ..... 1-5

**2. CONSTRUCT AND TAXONOMY OVERVIEW ..... 2-1**
2.1 CONSTRUCTS ..... 2-1

**3. NATIONAL IMAGERY TRANSMISSION FORMAT OVERVIEW ..... 3-1**
3.1 NITF FIELDS AND CHARACTER ENCODING ..... 3-1
3.2 NITF FILE STRUCTURE ..... 3-2
3.3 NITF FILE HEADER OVERVIEW ..... 3-3
3.4 NITF SEGMENTS ..... 3-5

**4. NITF – THE NITF FILE HEADER ..... 4-1**
4.1 NITF FIELDS ..... 4-1
4.2 NITF FILE HEADER ..... 4-4
4.3 NITF FILE HEADER – SEGMENT LIST ..... 4-19

**5. NITF – NITF FILE SEGMENTS ..... 5-1**
5.1 IMAGE SEGMENTS ..... 5-4
5.1.1 Image Data Masks ..... 5-17
5.2 GRAPHIC SEGMENTS ..... 5-20
5.3 TEXT SEGMENTS ..... 5-22
5.4 DATA EXTENSION SEGMENTS ..... 5-24
5.4.1 Traditional Data Extension Segments ..... 5-25
5.4.2 Overflow Data Extension Segments ..... 5-25
5.4.3 Streaming Header Data Extension Segments ..... 5-27
5.5 RESERVED EXTENSION SEGMENTS ..... 5-30
5.6 TAGGED RECORD EXTENSIONS ..... 5-32
5.7 ALL SEGMENT DATA ..... 5-33

**6. ACRONYMS ..... 6-1**

**7. REFERENCED DOCUMENTS ..... 7-1**

## LIST OF FIGURES

Figure 3-1. Transmitting a NITF File [1] ..... 3-1
Figure 3-2 NITF File Format [1] ..... 3-3
Figure 3-3. NITF File Header Introduction ..... 3-4
Figure 3-4. NITF File Header Security Classification Fields ..... 3-4
Figure 3-5. NITF File Header Declassification Fields ..... 3-4
Figure 3-6. NITF File Header Classification Authority ..... 3-4
Figure 3-7. Overall NITF File Header ..... 3-5
Figure 5-1. Display Level Example ..... 5-2
Figure 5-2. Blocked Image with Empty Block ..... 5-17
Figure 5-3. NITF with modified Image Mask ..... 5-19

# LIST OF TABLES

Table 1-1. Document Organization ..... 1-2
Table 1-2. Recommendation Actions ..... 1-3
Table 1-3. Recommendation Action Options ..... 1-4
Table 3-1. Character Sets in NITF ..... 3-2
Table 4-1 Example of Required and Conditional Header Fields ..... 4-3
Table 4-2. NITF File Header Fields ..... 4-5
Table 4-3. NITF File Header Example ..... 4-5
Table 4-4. Modified NITF File Header ..... 4-6
Table 4-5. NITF File Header Classification ..... 4-8
Table 4-6. File Classification Example ..... 4-9
Table 4-7. NITF File Declassification Fields ..... 4-12
Table 4-8. NITF File Header Classification Authority Fields ..... 4-14
Table 4-9. NITF File Header Originator Example ..... 4-17
Table 4-10. NITF File and Header Length Example ..... 4-18
Table 4-11. NITF File Header Segment Length Example ..... 4-21
Table 4-12. Data Hiding Changing Segment Count ..... 4-21
Table 5-1. Image Subheader Fields ..... 5-4
Table 5-2. NITF Image Subheader Example ..... 5-4
Table 5-3. Image Subheader Fields ..... 5-6
Table 5-4. Image Subheader – Comments Example ..... 5-8
Table 5-5. Image Compression Types ..... 5-9
Table 5-6. Image Band Representation Values ..... 5-11
Table 5-7. Image Mode Values ..... 5-14
Table 5-8. Image Mask Fields ..... 5-18
Table 5-9. Graphic Subheader Fields ..... 5-20
Table 5-10. Text Subheader Fields ..... 5-22
Table 5-11. Text Subheader Extended Data Fields ..... 5-23
Table 5-12. Data Extension Subheader Fields ..... 5-25
Table 5-13. Overflow Data Extension Subheader Fields (i) ..... 5-26
Table 5-14. Overflow Data Extension Subheader Fields (ii) ..... 5-26
Table 5-15. Data Extension Segment Subheader Fields for Streaming Header (i) ..... 5-27
Table 5-16. Data Extension Segment Subheader Fields for Streaming Header (ii) ..... 5-28
Table 5-17. Reserved Extension Segment Subheader Fields ..... 5-30
Table 5-18. RES User Defined Data Fields ..... 5-31
Table 5-19. Tagged Record Extension Data ..... 5-32
Table 5-20. TRE Data Example ..... 5-32
Table 6-1 Acronyms ..... 6-1

# 1. SCOPE

## 1.1 Purpose of this Document

The purpose of this document is to provide guidance for the development of a sanitization and analysis software tool for the National Imagery Transmission Format (NITF) Version 2.1. This document analyzes various elements and objects that are contained within the NITF file structure and then discusses the data hiding, data attack, and data disclosure risks. It describes how those elements can be a cause for concern for either hiding sensitive data or possibly attempting to exploit a system. This report provides numerous recommendations and mitigations that could be used to ensure the NITF file is safer and accurately conforms to the specification.

The intended audience of this document includes system engineers, designers, software developers, and testers who work on file inspection and sanitization applications that involve processing NITF files.

## 1.2 Introduction

File types that act as containers and store a variety of data introduce a significant amount of risk including hidden data, data disclosure, and data attack risks. Hidden data risks may be present where information can be stored within the file format and never presented to the end user. Complex file types can often be used to take advantage of vulnerabilities within applications; this requires the inspection of these files for correctness. NITF files contain images, graphics, text segments, and other user-defined text or binary data. The file format is designed to share imagery and associated data, while allowing applications using the format to select only the data items needed from a file [2]. Systems that create NITF files take image files, such as JPEG or JPEG 2000, as input and then transform them into a single NITF file. The NITF file may contain numerous images, overlays, graphics, symbols, and text. All the data is associated with each other inside a single combined file format; content can be overlaid on top of each other to provide markup or add text information to images. The NITF file can be shared with others whose applications can extract the content they need to display to the user.

## 1.3 Background

The main objective of NITF is "to provide a means for diverse systems to share imagery and associated data" [2]. NITF was first published in 1989 as version 1.1. When it

became a standard in 1991, the name changed to NITF Standard, or NITFS. NITF version 2.0 was completed in 1993. This document focuses on the latest NITF specification from MIL-STD-2500C (Version 2.1), published May 1, 2006. NITF files commonly use the ".ntf" or ".nitf" file extension. The NATO Secondary Image Format (NSIF) is commonly mentioned alongside NITF. NSIF is identical to NITF, except the first nine bytes of the file differ for versioning. The file extension is also typically ".nsf". This report does not explicitly address NSIF files, but since they are almost identical, this guidance could apply to this file format as well.

## 1.4 Document Organization

The following table summarizes the organization of this document.

**Table 1-1. Document Organization**

| Section | Description |
|---|---|
| Section 1: Scope | This section describes the scope, organization, and limitations of this document. |
| Section 2: Construct and Taxonomy Overview | This section describes the terms and constructs information for NITF. |
| Section 3: NITF Overview | This section describes the basic NITF file structure. |
| Section 4: The NITF File Header | This section describes the NITF File Header |
| Section 5: The NITF File Segments | This section describes all the data segments and subheaders following the NITF File Header. |
| Section 6. Acronyms | This section lists the acronyms that appear in this document. |
| Section 7: Referenced Documents | This section lists the sources that were used to prepare or cited in this document. |

## 1.5 Recommendations

The following subsections summarize the categories of recommendation actions that appear in this document and associated action options.

### 1.5.1 Actions

Each construct description lists recommended actions for handling the construct when processing a document. Generally, inspection and sanitization programs will perform

one or more actions on a construct: *Validate*, *Remove*, *Replace*, *External Filtering Required*, *Review, or Reject*.

The Recommendation section in each construct lists each of these actions and corresponding applicable explanations of the action to take. It notes if a particular action does not apply and indicates actions that are not part of the standard set of actions (listed in the previous paragraph). For example, a program may choose to reject a file if it is encrypted. Additionally, for some constructs, an action may further break down to specific elements of a construct (e.g., for hidden data in Excel, the recommendations for Remove include an action for removing hidden sheets and another for removing hidden cells) to give administrators the flexibility to handle specific elements differently.

**NOTE**

The recommendations in this document are brief explanations rather than a How-To Guide. Readers should refer to the construct description or MIL-STD-2500C for additional details.

Table 1-2 summarizes the recommendation actions.

**Table 1-2. Recommendation Actions**

| Recommendation Action | Comments |
|---|---|
| ***Validate*** | Verify the data structure's integrity, which may include integrity checks on other components in the file. (This should almost always be a recommended action) |
| ***Replace*** | Replace the data structure, or one or more of its elements, with values that alleviate the risk (e.g., replacing a user name with a non-identifying, harmless value, or substituting a common name for all authors). |
| ***Remove*** | Remove the data structure or one or more of its elements and any other affected areas. |
| ***External Filtering Required*** | Note the data type and pass the data to an external action for handling that data type (e.g., extract text and pass it to a dirty word search). |
| ***Review*** | Present the data structure or its constructs for a human to review. (This should almost always be recommended if the object being inspected can be revised by a human) |

| Recommendation Action | Comments |
|---|---|
| ***Reject*** | Reject the file. |

**NOTE**

No recommendations for logging all actions and found data are included here because all activity logging in a file inspection application should occur "at an appropriate level" and presented in a form that a human can analyze further (e.g., the audit information may be stored in any format but must be parsable and provide enough information to address the issue when presented to a human.)

## 1.5.2 Action Options

The companion to this document, *Data Transfer Guidance for NITF*, specifies four options for each recommended action: *Mandatory*, *Recommended*, *Optional*, or *Ignore*. Depending on the circumstances (e.g., a low to high data transfer versus a classified to unclassified transfer), programs can be configured to handle constructs differently.

Table 1-3 summarizes the recommendation action options.

**Table 1-3. Recommendation Action Options**

| Action Options | Comments |
|---|---|
| ***Mandatory*** | For the given direction (e.g., secure private network to unsecure Internet), the file inspection and sanitization program must perform this recommended action. |
| ***Recommended*** | Programs should implement this action if technically feasible. |
| ***Optional*** | Programs may choose to perform or ignore this recommended action. |
| ***Ignore*** | Programs can ignore this construct or data structure entirely |

### 1.5.3 Naming Convention for Recommendations

Recommendations in this document are numbered sequentially and can be identified by a number *x*, where *x* is a sequential number followed by the recommendation keyword defined in Table 1-2. There may be multiple recommendations of the same type (e.g., Validate) for any given finding, which remain uniquely identified by its number.

## 1.6 Data Transfer Guidance

Each format documented for inspection and sanitization programs has a companion document (i.e., the aforementioned DTG document). The DTG serves as a checklist for administrators and others to describe expected behaviors for inspection and sanitization programs. For instance, an administrator may decide to remove all hidden sheets in an Excel spreadsheet but leave hidden cells intact. Or, the administrator may decide to remove all hidden data if the document is being transferred to a lower security domain.

The DTG gives the administrator the flexibility to specify behaviors for inspection and sanitization programs. The workbook contains a worksheet for each security domain (i.e., the originating domain). Each worksheet lists the numbered constructs from this document and enumerated recommendations in a row. After the recommendations, the worksheet displays a cell for each possible destination domain. This enables an administrator to select the action option for data transfer from the originating domain to the particular destination domain. Each construct row also contains two comment cells: one for low to high transfers and another for high to low transfers.

The recommended actions address two broad risk types: data hiding and data execution. Most data structures are vulnerable to one risk type, while others are susceptible to both risk types. Each construct row in the DTG worksheet contains a cell for designating the risk type (i.e., data execution, data hiding, or both) and another cell for assessing the risk level for that construct (i.e., high, medium and low). This enables administrators to assign the risk type and risk level to each specific construct.

## 1.7 Document Limitations

### 1.7.1 Covert Channel Analysis

It is nearly theoretically impossible to detect or prevent covert channels during communication. It is impossible to identify all available covert channels in any file

format. Because NITF documents may contain free text and images, searching for hidden data becomes increasingly difficult. No tool can possibly analyze every channel, so this document highlights the highest risk areas to reduce or eliminate data spills and malicious content.

Additionally, this document does not discuss steganography within block text or media files, such as a hidden message that is embedded within an innocuous image or paragraph. Separate file format filters that specialize in steganography should be used to handle embedded content, such as text and images attempting to reduce the risk of steganography.

# 2. CONSTRUCT AND TAXONOMY OVERVIEW

## 2.1 Constructs

Although this document delves into many low level constructs in the NITF syntax hierarchy, it does not serve as a complete reference for all of the different constructs in the specification. This document focuses on particular areas of concern for developers of file inspection and sanitization programs; however, there are additional details in the standard that should be examined alongside this documentation. The constructs discussed are defined according to the format below. For each area of concern that is mentioned, the following sections exist:

- **Description**: a high level explanation of the data structure or element including an example if applicable.
- **Product**: provides the version or specification number where this construct exists.
- **Location**: provides a textual description of where to find the element in the document.
- **Concerns and Recommendations**:
  - **Concern**: An explanation of potential problems posed by the element is provided. For example, some metadata elements can cause inadvertent data leakage and others can be used for data exfiltration. A specific recommendation that follows can be used to address that concern.
  - **Recommendations**: As defined in Table 1-2.

Recommendations appear within each of the NITF constructs. For the purposes of this document, these recommendations are "alternatives." Some recommendations may seem better than others and some recommendations may be more difficult to implement. Certain recommendations complement each other and can be grouped together (e.g., "Replace field with default value" and "Pass free text to external filter"). Other recommendations may seem contradictory (e.g., "Remove field" and "Replace field").

# 3. NATIONAL IMAGERY TRANSMISSION FORMAT OVERVIEW

The National Imagery Transmission Format (NITF) is designed to share imagery and associated data [1]. The file format is composed of data segments including imagery, text, and graphics. NITF strives to provide comprehensive information contained entirely within a single file format that can meet the needs of a variety of users. NITF minimizes overhead and introduces simplicity into a file structure. NITF supports numerous types of associated data: images, graphics, text, miscellaneous data, and reserved areas for future content that has yet to be defined. In addition, there may be custom information exchanged within this file format in reserved areas. All this information, included within the file, provides a consolidated view of information in a common file format for use across different groups.

NITF is used as a container for many different types of imagery related data. Applications reading NITF files could present different types of imagery data combined into a single view. For example, images, text accompanying that image, and graphics or mark-up annotations related to the image can all be transmitted together via a single NITF container format. This format is shown in Figure 3-1, derived from a similar image in the NITF standard [1].

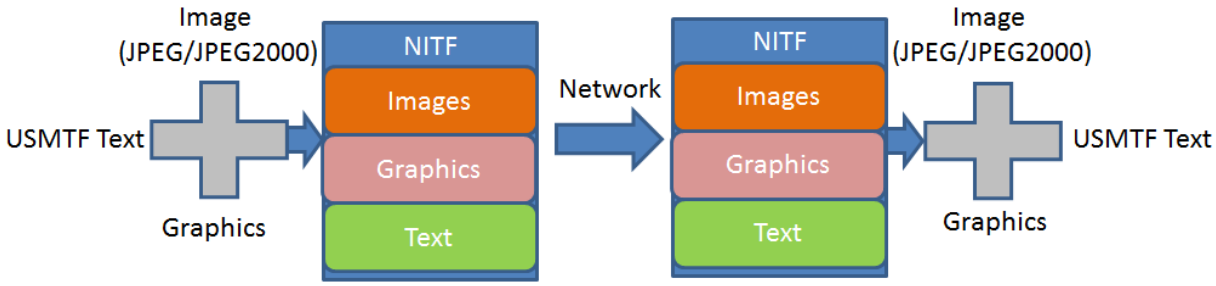


**Figure 3-1. Transmitting a NITF File [1]**

## 3.1 NITF Fields and Character Encoding

The NITF standard defines numerous character sets, which provide a fixed group of characters that are permitted. Fields and values defined in the NITF specification specify which character set shall be used for a particular file. The following table defines each character set defined by the NITF standard. This report will reference these terms in later sections.

**Table 3-1. Character Sets in NITF**

| Term | Definition | ASCII Character Range |
|---|---|---|
| BCS | Basic Character Set | 0x20 – 0x7E,<br>plus<br>0x0A (Line Feed),<br>0x0C (Form Feed),<br>0x0D (Carriage Return) |
| BCS-A | Basic Character Set – Alphanumeric | 0x20-0x7E |
| BCS-N | Basic Character Set – Numeric Integer | 0x30-0x39,<br>plus<br>0x2B (Sign Code),<br>0x2D (Minus Sign) |
| BCS-N Positive Integer | Basic Character Set – Numeric Positive Integer | 0x30-0x39 |
| ECS | Extended Character Set | 0x20-0x7E,<br>0xA0-0xFF,<br>plus<br>0x0A (Line Feed),<br>0x0C (Form Feed),<br>0x0D (Carriage Return) |
| ECS-A | Extended Character Set – Alphanumeric | 0x20-0x7E,<br>0xA0-0xFF |
| ECS Space/BCS Space | Extended Character Set/Basic Character Set Space | 0x20 |

## 3.2 NITF File Structure

The file structure of NITF Version 2.1 is shown below in Figure 3-2. It is comprised of an NITF File Header followed by optional segments of data. There can be numerous segments of each type. The number of segments is defined by a required field in the NITF header. Each segment is then comprised of a subheader followed by a block of data. Per the NITF standard, "It is possible to include zero, one, or multiples of each

standard data segment in a single file (for example: several images, but no graphics)" [1]. The NITF File Header indicates how many segments, and the length of each subheader and its segment data in the file. The file is length delimited by these values in the NITF File Header. If these values are incorrect, there will be problems parsing the data later in the file. If fewer segments are specified than actually exist then data could exist in the file that is ignored. Each subheader implements the first two bytes as a marker that indicates the start of a particular segment. These values should be carefully checked to ensure that the parser is expecting the beginning of a new segment. Although the figure below indicates a Reserved Segment between Graphic and Text segments, as of the current standard, there is nothing defined for this location. Currently, Text segments immediately follow Graphic segments.

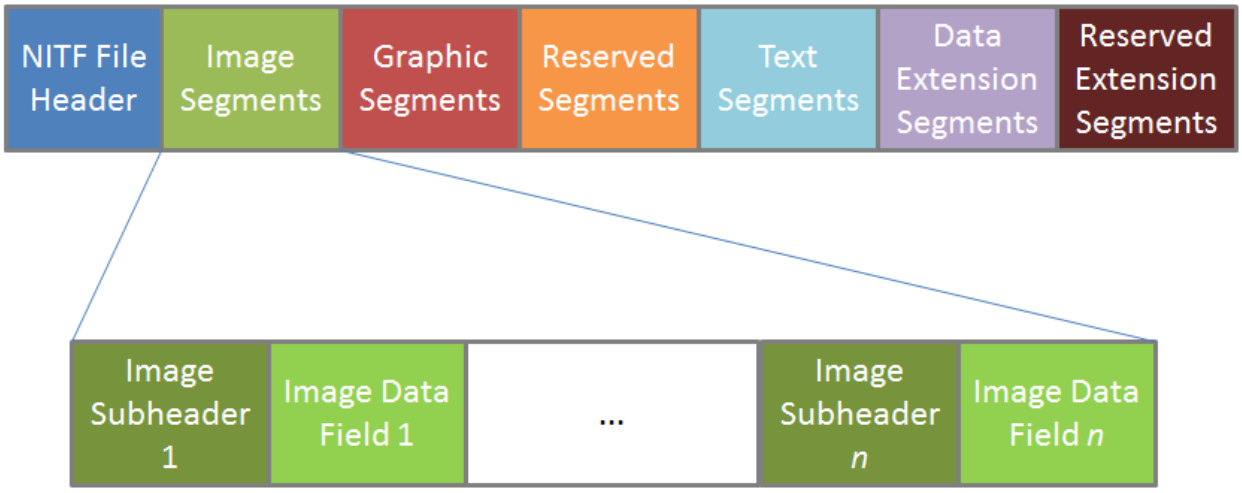


**Figure 3-2 NITF File Format [1]**

## 3.3 NITF File Header Overview

The NITF File Header starts at the beginning of the file and ends just before the start of the first Image Segment Subheader. The first few bytes of the NITF header have a structure similar to Figure 3-3 below. It includes the magic number of NITF, which is a four byte field at the very beginning of the file used to identify the file format. Many different file types reserve the first few bytes of the file to define the file type, commonly referred to as a magic number. There are other fields in the NITF File Header such as the version of NITF, date and time, and title information are also present. Field lengths and default values are discussed in the next section.


**Figure 3-3. NITF File Header Introduction**

The file header fields are followed by fields describing the File Security Classification, Control and Handling, and Release Instructions of the file. This information is derived from the NITF Specification in Table A-1 of the Appendix.

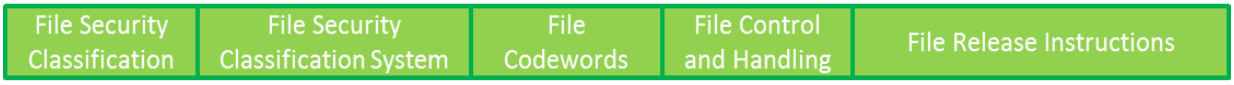


**Figure 3-4. NITF File Header Security Classification Fields**

The next set of fields describes declassification or downgrade instructions for this file. Dates or events can be established in this part of the header to indicate how to handle the declassification of the NITF file if triggered by a date or event. Figure 3-5 shows the layout of each of these fields in the NITF File Header. These fields are described in more detail in Section 4.

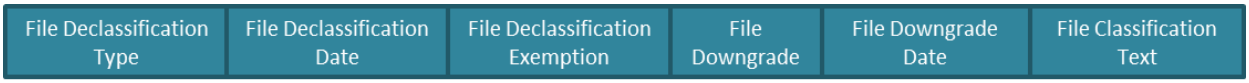


**Figure 3-5. NITF File Header Declassification Fields**

The next fields describe the File Classification Authority, the reason for classification, the file security source date, and the control and copy number. The following image in Figure 3-6 illustrates the order of these fields in the NITF File Header.


**Figure 3-6. NITF File Header Classification Authority**

There are several more fields that are addressed in later sections of this report. When combined with the fields discussed earlier, the overall NITF File Header resembles the image in Figure 3-7. This image is for illustration purposes only and does not accurately show the comparative byte length of each field.

In Figure 3-7, the grey fields on the lower right are conditional fields and may not exist in the file. In addition, there may be 0 or more of each segment type indicated within

the defined Header field. For example, the Number of *Image* Segments is a value that indicates how many pairs of *Image* Subheader length and Image Data Length pairs of fields that follow (shown in grey in Figure 3-7). This allows NITF viewers to determine the size and location of each block of data that follows the NITF File Header. The User Defined Header and Extended Header are the means for adding additional data that is not part of the segment data located later in the file.

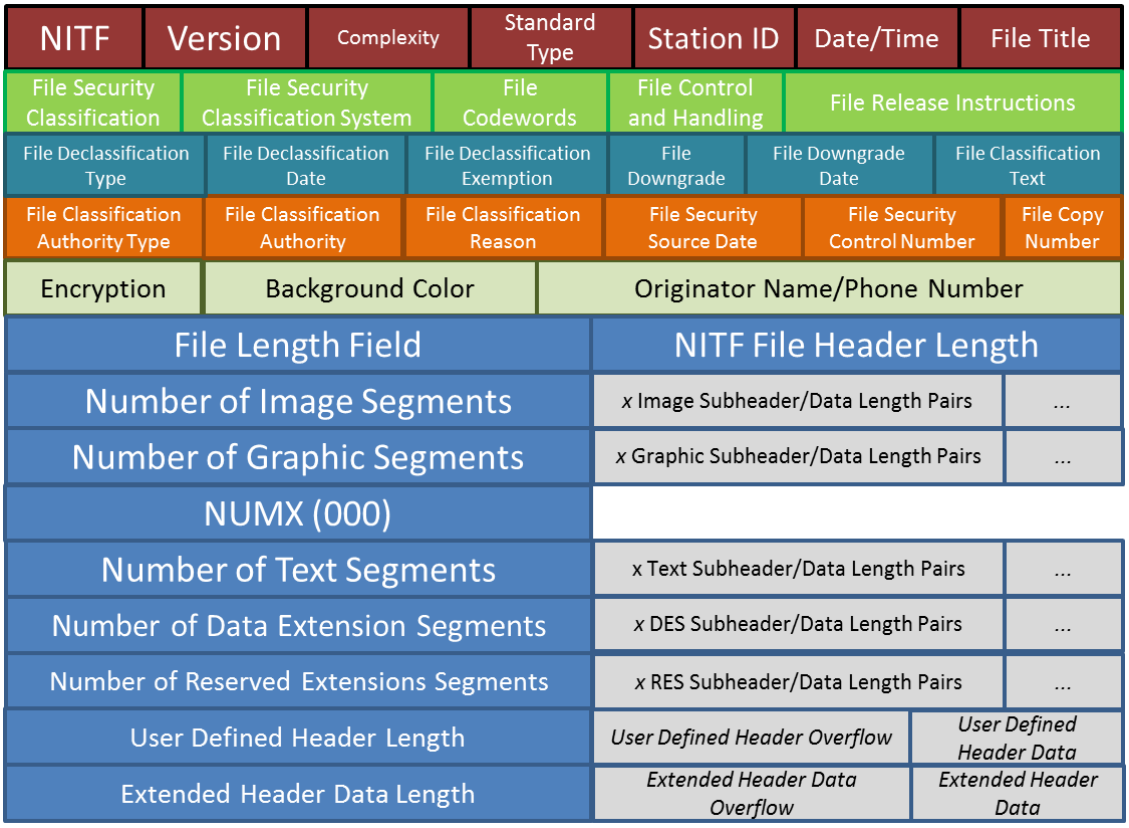


**Figure 3-7. Overall NITF File Header**

## 3.4 NITF Segments

There are 5 types of segments in the NITF file: Image, Computer Graphics (Computer Graphics Metafile), Text, Data Extension Segment (DES), and Reserved Extensions Segment (RES). As discussed previously, a NITF file may contain 0 or more of each type of segment. The NITF file header describes how many segments exist in the file, the length of each segment's subheader, and the actual length of the segment data. The

NITF File Header is not fixed in length and depends on how many segments are defined in the file.

Each NITF segment is comprised of a subheader and the data portion of the segment. The subheader provides additional information such as classification details related to the individual segment and other information to aid in parsing the data segment. Each subheader and data portion is discussed in further detail in Section 5.

# 4. NITF – THE NITF FILE HEADER

## 4.1 NITF FIELDS

Before defining the NITF file header in more detail, this section introduces the two different types of fields in NITF: Required and Conditional. Both of these types of fields appear in the NITF File Header and every subheader. Required fields are always present in the file; however, they may not always be implemented or contain a value. Examples of this are discussed in Section 4.2. Most default values for these required but unimplemented fields are all spaces (0x20), but all default values are defined in the NITF standard. Conditional fields are only present in the file if the values in other required fields indicate they should be present. If the condition is not met, the field simply does not exist. If conditional fields are improperly added into the file, a parsing error may result later in the file when validating a field's value. This section demonstrates that all fields within NITF, including fields in subheaders defined in later sections, can be inspected to a certain degree. This is because all fields in NITF have been defined with a character set; some have minimum and maximum values, while others may be defined as a constant.

### NITF.4.1: ALL REQUIRED FIELDS

**DESCRIPTION:**

The NITF specification defines several required fields both in the File Header and each subheader. They are denoted in the specification through the value of TYPE 'R' in the Appendix of the NITF standard. There are also some fields with TYPE equal to '<R>', which means that if they are required fields, but their value is optional and may contain all spaces (0x20). These fields exist in all NITF files, whether they contain spaces or actual valid content. Appendices in the NITF standard define the "VALUE RANGE" for each field. The value range specifies the characters set such as BCS-A, BCS-N, or ECS-A, all are defined in Table 1-4 of this report. All lengths for each field are also defined by the NITF standard in Appendix A. There are a few variable length fields in NITF files, which have the length defined in a previous field.

All NITF fields are defined with a size field, which allows for each field to be mapped to a specific byte offset within the file. Although it may be difficult to determine when one field ends and another begins, an invalid character in a field may trigger this discrepancy. Numerical fields that do not take up the entire the field length are preceded with zeros. Other fields that do not required the entire field length often contain spaces (0x20) following the end of the field's content.

In the Appendix of the NITF specification, some fields are provided with a minimum and maximum value or a range of defined values. If a field has an incorrect value it is primarily due to an invalid writer application. This section serves as a catch-all in cases where specific fields

are not mentioned in later sections. Required fields mentioned later in this document have further guidance for inspection.

**Example:** The first three fields of the NITF File Header:

"NITF02.1001"

| Header Field | Value | Comments |
|---|---|---|
| File Profile Name | NITF | Size: 4<br>Character Set: BCS-A<br>Must be NITF |
| FVER (File Version) | 02.10 | Size: 5<br>Character Set: BCS-A |
| CLEVEL (Complexity Level) | 01 | Size: 2<br>Character Set: BCS-N<br>Min: 01<br>Max: 99 |

## PRODUCT: NITF 2.1

### LOCATIONS:

Required fields are located in the NITF field header as well as numerous subheaders.

### CONCERNS AND RECOMMENDATIONS:

**Data Hiding:** A required field may not be parsed by the file viewer, in the case of metadata or required fields that are unimplemented (<R>). Other required fields may simply be ignored by some viewers. If characters other than the default reside in these locations, it may introduce a hidden data risk.

1 **Validate**: Check for each required field, that every character in that field belongs to its corresponding character set defined in the Appendix of the NITF standard or contains a reasonable value passing a regular expression.

2 **Validate**: If a minimum and maximum value is defined in the standard, check that the value provided in all required fields falls within that range.

3 **Validate**: If a field has a specific value or constant defined by the standard, ensure that it is correct according to Appendix of the NITF standard.

4 **Replace**: If the type is listed as <R> and the field is not used, replace the contents with its default value or all spaces (0x20). It may not be possible to determine if a field is used, for example, the Field File Title is listed as <R> but is simply metadata for the file. Other fields are listed as <R> but the standard indicates that they must contain a value depending on values in other Required fields in the Header.

**5** **Replace**: If the field is reserved (possibly for future use as some are listed in NITF Appendix) and has no purpose at this time, replace the value with its default value from the standard.

## NITF.4.1: END

## NITF.4.2: ALL CONDITIONAL FIELDS

### DESCRIPTION:

Conditional fields are defined in the specification with the Type 'C', as opposed to Type 'R' or '<R>' for required. They are conditional because the existence of the field depends on values in other required fields.

For example, a NITF file can contain anywhere between 0 to 999 Image Segments. The number of image segments is a required field in the NITF header, which should be inspected according to NITF.4.1. If the value is greater than 0, there should exist a length field for each image subheader and image segment in the NITF header. These two fields are conditional, but are defined with a character set and minimum and maximum values, just as required fields.

**Table 4-1 Example of Required and Conditional Header Fields**

| NITF Header Field | Value | Comments |
|---|---|---|
| Number of Image Segments (NUMI) | 003 | REQUIRED: There are 3 image segments<br><br>Character Set: BCS-N (Positive Integer)<br><br>Range: 000-999 |
| Length 1st Image Subheader (LISH001) | 000500 | CONDITIONAL: The length of subheader for the first segment.<br><br>Character Set: BCS-N (Positive Integer)<br><br>Range: 000439-999998 |
| Length of 1st Image Segment (LI001) | 0003000000 | CONDITIONAL: The length of First Image segment.<br><br>Character Set: BCS-N (Positive Integer)<br><br>Range: 0000000001-9999999998 |

Invalid conditional fields are difficult to detect since they may contain characters sets of other adjacent fields. If a parsing error or invalid character set or value is detected in a later field, it may be the result of an invalid conditional field (a conditional field that is present when it should not be).

### PRODUCT: NITF 2.1

### LOCATIONS:

Conditional fields are located in the NITF field header as well as numerous subheaders.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding and Data Attack:** If conditional values are not present or have incorrect values, segments of data could possibly go undetected. Crafted values in fields could be used as a data attack on the parsing of the file.

1 **Validate**: Check for each conditional field, if it exists, that every character in that field belongs to its corresponding character set defined in the Appendix of the NITF standard.

2 **Validate**: If a minimum and maximum value is defined in the specification, check that the value provided in all required fields falls within that range.

3 **Validate**: For each conditional field, check the conditional requirements are correct. For example, check that each conditional field is accurate per the number of each segments, and that each segment exists later in the file.

4 **Remove**: If the existence of a conditional field can be determined, remove conditional fields that do not meet their requirements. This may require re-aligning or rebuilding the file.

**NITF.4.2: END**

## 4.2 NITF FILE HEADER

The NITF File Header is the first few hundred bytes in the file. It is variable in length because a NITF file may contain multiple segments. The length of each segment subheader and segment data are defined in the NITF File Header, so parsers can extract the necessary data from the file. The NITF File Header contains many other fields as well that provide metadata, such as file security classification, the background color, and release instructions. All of these fields are discussed in constructs in this section.

## NITF.4.3: NITF HEADER

### DESCRIPTION:

The NITF header includes at a minimum the first 480 bytes of the file. This includes metadata information about the file before defining the number and length of each type of segment. This subsection focuses only on the first 119 bytes of the file which contain the first 7 fields of information. The remainder of the NITF header is discussed in later subsections of Section 4. The standard defines each of these fields, the character set for them, and their length. The parsing application assumes that each field is of the proper length. If this is not the case due to an invalid writer application, then a parsing error will likely be discovered in later fields.

The first seven fields of the NITF header are defined as follows:

**Table 4-2. NITF File Header Fields**

| Field | Size (in bytes) | Description |
|---|---|---|
| File Profile Name (FHDR) | 4 | Shall be equal to "NITF" (Note for NSIF Files: "NSIF") |
| File Version (FVER) | 5 | A string defining the version. A valid example is "02.10", for NITF 2.1 version. (Note for NSIF Files: "01.00") |
| Complexity Level (CLEVEL) | 2 | Complexity level of the file required to parse the remainder of it. Allowed values are defined in specification in Table A-10 of the NITF specification. |
| Standard Type (STYPE) | 4 | Shall be equal to BF01, which indicates the file is formatted using ISO/IEC IS 12087-5. |
| Originating Station ID (OSTAID) | 10 | The identification code or the name of the station originating the content. |
| File Date and Time (FDT) | 14 | Date and Time in UTC format. |
| File Title (FT) | 80 | A string of 80 bytes for the file title. |

Example from NITF Specification:

**Table 4-3. NITF File Header Example**

| Field | Value |
|---|---|
| File Profile Name | NITF |
| File Version | 02.10 |
| Complexity Level | 05 |
| Standard Type | BF01 |
| Originating Station ID | U21SOO90 |
| File Date/Time | 19960930224632 |
| File Title | MAJOR TEST FACILITY |

Modified example with comments addressing changes:

**Table 4-4. Modified NITF File Header**

| Field | New Value | Comments |
|---|---|---|
| File Profile Name | `NITF` | This must be NITF or the file will not open by image viewers. |
| File Version | `74.56` | Image viewers show no impact when field is incorrect. |
| Complexity Level | `99` | Image viewers show no impact on basic image files when field is incorrect. |
| Standard Type | `xxxx` | Image viewers show no impact when field is incorrect. |
| Originating Station ID | `DIFFERENTx` | Image viewers show no impact when field is incorrect. |
| File Date/Time | `It's April 1st` | Image viewers show no impact when field is incorrect. |
| File Title | `Here is place for free text, anything may reside in this location, you have 80B.` | Image viewers show no impact since it's free text. |

### PRODUCT: NITF 2.1

#### LOCATIONS:

The fields discussed in this section are located in the first 119 bytes of the file.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding and Data Attack:** From experimenting with NITF viewers, the only field required to be accurate is the File Profile Name, which should be "NITF". The next bytes of this header portion did not appear to have a significant impact on the file when opening it so they may contain hidden data or malicious content. The Complexity Level may have an impact on certain images and applications.

**1** **Validate**: Check that the first 4 bytes of the file are NITF.

**2** **Validate**: Check that the File Version field is within the defined allowable values; for example, 02.10 is defined by the latest specification as valid.

**3** **Validate**: Check that the complexity level field is equal to one of the values defined in Table A-10 of the NITF Specification Appendix. It should be equal to 3, 5, 6, 7, or 9.

**4** **Validate**: Check that the date field is in the correct UTC format of CCYYMMDDhhmmss.

**5** **Replace**: Replace fields with default values, if possible. This may impact end users, if they require this information.

**6** **Reject**: Reject files that do not contain NITF in the first 4 bytes.

**Data Disclosure:** The NITF header contains the fields Originating Station ID, File Date and Time, and the File Title. All three of these fields could contain sensitive information.

**7** **External Filtering Required**: Pass the contents of each field to an external filter as they may be free text.

**8** **Review**: Manually review the contents of File Title and Station ID.

## NITF.4.3: END

## NITF.4.4: NITF HEADER – FILE SECURITY CLASSIFICATION

### DESCRIPTION:

Following the first seven fields of the NITF header is information regarding the security classification of the file. There are a total of 5 fields that are included in this section. The following table describes each of these fields, their length, and expected values. If any of these fields are unused, they default to space (0x20).

The first field is a 1 byte security classification identifier. If the value is U for unclassified, the next 15 fields are left as spaces, since they are unused but still Required (<R>). If implemented, the following fields are present to provide more information. This is a conditional check on a required field that should be made.

**Table 4-5. NITF File Header Classification**

| Field | Length (bytes) | Description |
|---|---|---|
| File Security Classification | 1 | Either U, T, S, C, or R. |
| File Security Classification System | 2 | Contains values for national or multinational security systems as defined in FIPS PUB 10-4 (according to MIL-STD-2500C) or the newer Geopolitical Entities, Names, and Codes (GENC) standard. If the File Security Classification is 'U', then this field is all spaces.<br><br>It is listed as a <R> required field, but may contain spaces depending upon the classification. This applies for all fields following the File Security Classification. |
| File Codewords | 11 | A list of values of codewords that apply to this file. Each codeword is defined in Table A-4 of the NITF specification and is separated by a space (0x20). |
| File Control and Handling | 2 | Security control and handling instructions, defined by Table A-4 of the NITF specification. |
| File Releasing Instructions | 20 | Contains a list of countries for which file is authorized for release, each separated by a space (0x20). Valid list of values is located in FIPS PUB 10-4. |

Simple Unclassified Example (Single quotes are shown for illustration only)

**Table 4-6. File Classification Example**

| Field | Value | Description |
|---|---|---|
| File Security Classification | U | Unclassified |
| File Security Classification System | ' ' | 2 spaces (0x20), no single quotes |
| File Codewords | ' ' | 11 spaces (0x20), no single quotes |
| File Control and Handling | ' ' | 2 spaces (0x20), no single quotes |
| File Releasing Instructions | ' ' | 20 spaces (0x20), no single quotes |

**PRODUCT: NITF 2.1**

**LOCATIONS:**
These 5 fields begin at Byte 120 and end at Byte 154.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding and Data Disclosure -** If these fields are not implemented, the specification indicates that the values should be all spaces (0x20). This is an area for free text or information to hide since they could be ignored if they are not implemented. If these fields contain valid information, it could be an accidental disclosure risk, since it will not appear as file content.

**1** **Validate**: Check that the File Security Classification field is a valid value: U, T, S, C, or R.

**2** **Validate**: If FSCLAS is T, S, C, or R, then FSCLSY must not be spaces. If any of the following fields are populated with anything other than spaces FSCLSY must not be spaces: FSCODE, FSREL, FSDCTP, FSDCDT, FSDCXM, FSDG, FSDGDT, FSCLTX, FSCATP, FSCAUT, FSCRSN, FSSRDT, and FSCTLN.

**3** **Validate**: Check that the File Security Classification System contains a valid 2 byte field. This field may be defined in either FIPS PUB 10-4 or the Geopolitical Entities, Names, and Codes (GENC) standard, which defines the latest set of country codes.

**4** **Validate**: Check that the File Codewords are all from Table A-4 in the NITF Specification, and if there are multiple codewords, that they are separated by a space (0x20). Also consult with current security directives when implementing, as this table may be out of date.

**5** **Validate**: Check that the File Control and Handling field is a valid 2 byte value from Table A-4 of the NITF specification. Also consult with current security directives when implementing, as this table may be out of date.

**6** **Validate**: Check that the File Releasing Instructions field contains valid entries from Federal Information Processing Standard (FIPS) PUB 10-4 or GENC standard, and if multiple, that they are separated by a space (0x20).

**7** **Replace**: If FSCLAS is U and FSCLSY is spaces (0x20) replace the following fields with spaces: FSCODE, FSREL, FSDCTP, FSDCDT, FSDCXM, FSDG, FSDGDT, FSCLTX, FSCATP, FSCAUT, FSCRSN, FSSRDT, and FSCTLN.

**8** **Review**: Manually review each field in this header with values from Table A-4 (NITF), FIPS PUB 10-4, or GENC to ensure that they are correct.

**9** **External Filtering**: If fields contain valid values of text, pass to an external filter.

**10** **Reject**: Reject NITF files that contain invalid classification values.

## NITF.4.4: END

## NITF.4.5: NITF HEADER - DECLASSIFICATION INFORMATION

### DESCRIPTION:

Following the fields described in NITF.4.4, the next 6 fields are related to file declassification instructions. This table describes these fields and their possible values. If the values defined below are less than the full length of the field, the rest of the field is spaces (0x20). Parsing applications should handle removing the whitespace around fields if it exists.

**Table 4-7. NITF File Declassification Fields**

| Field | Length (bytes) | Description |
|---|---|---|
| File Declassification Type | 2 | Valid values are<br>DD (Declassify on Date),<br>DE (Declassify on Event),<br>GD (Downgrade on Date),<br>GE (Downgrade on Event),<br>O (OADR), or<br>X (Exempt).<br>Default is 2 spaces. |
| File Declassification Date | 8 | Date of declassification only if the field above is DD. In the format CCYYMMDD. |
| File Declassification Exemption | 4 | The reason the file is exempt, only if the declassification type is X. Valid values are X1-X8 (see paragraph below) and X251-X259. If not implemented, then 4 spaces are used as a default.<br><br>NITF Files originating after September 22, 2003 are not to use codes X1-X8, except when deriving information from a source using those codes. |
| File Downgrade | 1 | Valid values are S, C, or R. Default is space if unused. |
| File Downgrade Date | 8 | If declassification type is GD, then a date is supplied in the format CCYYMMDD. |
| File Classification Text | 43 | If declassification type if DE or GE, this field contains free text of additional information. |

### PRODUCT: NITF 2.1

### LOCATIONS:

These fields start at Byte 155 in the NITF file and continue until Byte 220.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding –** If these fields are not implemented, the standard indicates that the values to be all spaces (0x20). This is an area for free text or information to hide since they could be ignored.

**1** **Validate**: Check that the File Declassification Type is a valid value as defined in the table above.

**2** **Validate**: If the File Declassification Type is DD, ensure that the File Declassification date contains a value (not spaces).

**3** **Validate**: If File Declassification date is present, ensure that it formats to CCYYMMDD.

**4** **Validate**: If File Declassification type is X for Exempt, check that the File Declassification Exemption value is a valid (Range is X1-X8 (if applicable from Table 4-7) and X251-X259).

**5** **Validate:** If File Declassification Type is one of two downgrade options GE or GD, check that the File Downgrade value is S, C, or R.

**6** **Validate:** If File Declassification Type is GD, check that the File Downgrade Date is implemented as CCYYMMDD.

**7** **Replace:** Replace all fields with all spaces only if the previous construct identified an unclassified file.

**8** **Replace:** Replace all fields with all spaces if the File Declassification Type is not implemented (all spaces).

**Data Disclosure –** Certain fields contain free text of information, which might accidently contain sensitive information.

**9** **External Filtering Required**: Pass the free text in File Classification Text for external review.

**NITF.4.5: END**

## NITF.4.6: NITF HEADER – FILE CLASSIFICATION AUTHORITY

### DESCRIPTION:

The NITF header contains information regarding the type of authority that was used to classify the file. This follows the fields after NITF.4.5. The fields, length, and a description are shown in the following table.

**Table 4-8. NITF File Header Classification Authority Fields**

| Field | Length (bytes) | Description |
|---|---|---|
| File Classification Authority Type | 1 | Valid values are O, D, and M. |
| File Classification Authority | 40 | Free text values to identify the classification authority. |
| File Classification Reason | 1 | Valid values are A-G, defined per E.O. 12958, Section 1.5(a) to (g). Other valid values are H, M, and N. Code "M" designates two or more classification reasons apply, which shall be listed in the File Classification Text field presented in construct NITF.4.5. |
| File Security Source Date | 8 | Contains the date of the source from which this it was derived. In the format CCYYMMDD. |
| File Security Control Number | 15 | Contains a security control number for the file. |
| File Copy Number | 5 | Contains the copy number of file. The default is all zeros which indicate there is no tracking for copies of the file. |
| File Number of Copies | 5 | Contains the total number of copies of the file. The default is all zeros which indicate there is no tracking of numbered file copies [1]. |

### PRODUCT: NITF 2.1

#### LOCATIONS:

The fields in this section begin at Byte 221 and end at Byte 295 in the NITF header.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding** – If these fields are not implemented, then hidden data may reside here since they will be ignored.

**1** **Validate**: Check that the File Classification Authority type is O, D, M, or an ECS space (0x20).

2 **Validate:** Check that the File Classification Reason is within the range of A-G, H, M, and N or contains only an ECS space (0x20).

3 **Validate:** Check that the File Security Source Date is in the format of CCYYMMDD or consists entirely of ECS spaces (0x20).

4 **Validate:** Check that the Copy and Number of Copies fields are actual numbers.

5 **Validate:** Check that File Copy Number is less than or equal to File Number of Copies.

**Data Hiding and Data Disclosure –** Free text values within the File Classification Authority field may introduce a hiding and a disclosure risk.

6 **External Filtering Required**: Pass the File Classification Authority free text to an external filter.

## NITF.4.6: END

## NITF.4.7: ENCRYPTION

### DESCRIPTION:

The NITF 2.1 File header supports a single byte to indicate an encrypted file. This field follows the last construct in the byte ordering from NITF.4.6. Its definition is simple: a 0 is used if no encryption, which is default, and a 1 is used if the file is encrypted. At the moment, encryption is not supported, so this value should always be 0 until the specification changes.

Each Subheader contains this field as well for encrypting individual segments, which is unsupported at this time. This construct applies to all instances of the Encryption field located throughout the entire NITF File, not just in the NITF File Header. There could be up to 4,995 segments in the file, since the maximum number of each segment is 999, and there are five different types of segments. This is possible as long as the maximum NITF File Size is not reached. With that number of unused Encryption Fields there could be thousands of single unused encryption bytes spread across the file.

### PRODUCT: NITF 2.1

### LOCATIONS:

The Encryption byte in the NITF File Header is at Byte 296. Each subheader for each segment contains a single byte Encryption field following the classification fields (similar to NITF File Header).

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding** - Since this field may exist across numerous segments of data, the combination of all encryption bytes can lead to a hiding risk, with 4,996 bytes where information can be hidden.

**1** **Validate**: Ensure that this value is equal to 0.

**2** **Replace**: Replace any value in this field to a 0.

**3** **Reject**: Reject files that claim to be encrypted, as of NITF 2.1, neither an entire NITF file nor any of its individual segments can be encrypted.

## NITF.4.7: END

## NITF.4.8: NITF HEADER - BACKGROUND COLOR

#### DESCRIPTION:

NITF 2.1 introduced a field in the NITF header to define the background color in an RGB format. This was not present in previous versions of the NITF specification. The specification defines this value as the color of the background that is behind displayable segments [1]. The specification also indicates that this field is present to eliminate the possibility to "lose information if the originator selects a presentation color that is the same as the receiver's selected background color" [1].

#### PRODUCT: NITF 2.1

#### LOCATIONS:

The background color field in the NITF File header is located at Byte 297 and is 3 bytes long.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding –** Other layers present in NITF may be the same color as the background color which may introduce a hidden data risk. This doesn't appear to occur with an image as the image is overlaid on top of the background, but text may have the same color as the background.

**1** **Validate**: Check that the background color does not mask with other NITF content.

**2** **Replace**: Change the RGB color value such that it is different than elements.

**3** **Review**: Manually change the background color on images to other values to ensure no data is hidden.

## NITF.4.8: END

## NITF.4.9: NITF HEADER - ORIGINATOR INFORMATION

### DESCRIPTION:

The NITF Header supports including Originator information. The first construct introduced the Originating Station ID. This section covers the two fields following the File Background Color. The two fields are the Originator's Name and the Originator's Phone Number. The specification indicates that these fields may be left blank as they are not necessary.

**Table 4-9. NITF File Header Originator Example**

| Field | Value |
|---|---|
| Originator's Name | John Doe |
| Originator's Phone Number | (800) 555-1212 |

### PRODUCT: NITF 2.1

#### LOCATIONS:

The Originator Name field starts at Byte 300 and is 24 bytes long. The Originator Phone field starts at Byte 324 and is 18 bytes long.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding –** Data could be inserted in both fields knowing that it might not be parsed by an application.

**1** **Validate**: Check to ensure that the Originator's Name value is on an approved whitelist.

**2** **Validate**: Check that both fields match an approved whitelist or approved list of regexes to ensure that they are valid names and numbers.

**3** **Validate**: Check against a location's phone list to ensure that the Originator's Phone Number is for that user or location and is approved for release.

**Data Disclosure –** Accurate information in these fields may introduce an accidental data disclosure leak.

**4** **Replace**: Replace these two fields with all spaces, they are not needed.

**5** **Replace**: Replace these two fields with a releasable Point of Contact's (POC) information.

6 **External Filtering Required**: Pass both the contents of these fields to an external filter.

## NITF.4.9: END

## NITF.4.10: NITF HEADER - FILE LENGTH AND HEADER LENGTH

### DESCRIPTION:

The final two fields of the NITF header are the File Length field and the NITF File Header Length field. Both of these fields are important, and if they are incorrect, viewers may not open the file. The File Length is the length of the entire file including headers, subheaders, and all data.

The NITF File Header contains all of the fields discussed in Section 4.2. As later constructs show, the size can be variable depending on the number of segments in the data portion of the NITF file.

The specification indicates that the File Length field may not be known when the header is created. In this case, there is a special reserved Length field. The File Length field is 12 bytes long, and a value of 999999999999 indicates an unknown size at the time of creation.

This is an example of a 1.049479MB file with a 404 byte header.

**Table 4-10. NITF File and Header Length Example**

| Field | Value |
|---|---|
| File Length Field | 000001049479 |
| NITF File Header Length | 000404 |

### PRODUCT: NITF 2.1

#### LOCATIONS:

The File Length field begins at Byte 342 and is 12 bytes long. NITF Filer Header Length fields starts at Byte 354 and is 6 bytes long.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack –** Although not a documented attack, any concern with reading length values may introduce a data attack since applications may rely on these being accurate without first trying to verify its correctness.

1 **Validate**: Check that the File Length field is the actual size of the entire file. The range should be between 388 and 999999999999. If the value is all 9's, the actual File Length field will be in the streaming DES subheader.

2 **Validate**: Check that the NITF File header length is the actual length. This will involve parsing the remainder of the header to determine the number of segments and when the

NITF file header terminates. The range should be between 388 and 999999. Again, if this field is all 9's, the actual Header Length will be in the streaming DES subheader.

**3** **Replace**: Replace the File Length field with the actual size of the entire file if incorrect and can be determined at that time.

**4** **Replace:** Replace the NITF File Header Length field with the actual size of the NITF File Header.

**5** **Reject:** Reject files with incorrect file or header lengths.

**NITF.4.10:END**

## 4.3 NITF File Header – Segment List

The NITF File Header introduces the different types of segments that exist in the file. A NITF file may contain 0 or more Image Segments, Graphic Segments, Text Segments, Data Extension Segments, Reserved Extension Segments, User Defined Header and data, and extended header data. All blocks of data in a NITF file are covered in Section 5.7 since they are outside the scope of NITF. The NITF File Header defines how many segments and the length of each block of data.

Segments are defined in the NITF File header by a field that indicates how many segments of each type. This field is immediately followed by a pair of fields that indicate the length of each segment subheader and length of actual segment data. This repeats for each segment that exists in the file. In order to ensure that all fields are implemented correctly, each field shall be parsed along with the remainder of the file to ensure that everything is aligned and correctly sized. Since all of the length fields discussed in this section are numbers, it will be difficult to determine if each field is correct until parsing all of them and the rest of the file contents.

## NITF.4.11:NITF HEADER – ALL SEGMENT DEFINITIONS

### DESCRIPTION:

Following the NITF File Header Length field is the Number of Image Segments Field (NUMI). This is a required field and may be anywhere between 0 and 999. If this value is greater than 0, then there is an additional pair of fields called "Length of $n^{th}$ Image Subheader" (LISHn) and "Length of $n^{th}$ Image Segment" (LIn). These are both Conditional fields as discussed in NITF.4.2, since the NUMI field must be greater than 0 for these fields to exist. For example, if there are 3 image segments, then following the NUMI field, there should be 3 pairs of LISHn and LIn fields, 7 fields total, described the image segments.

The LISHn fields may contain values ranging from 000439 to 999998. Like other length values, a special reserved length of 999999 is used when the size of the image segment is not known when the header is created. This is referred to as a streaming file. In a streaming file, the complete NITF header follows the data in a Data Extension Segment (DES). The complete NITF header contains the actual length that was determined after writing the image to the file. This is covered in Section 5.4.3.

LIn fields may contain values ranging from 0000000001 to 9999999998. Again, a special reserved value of 9999999999 is used when the image length is not known when creating the header.

Table A-1 of the NITF specification defines the ranges of each field defined in this construct. This should be referenced when checking each field.

The next segment definition is for Graphics and is provided in the header in the same manner as Images. A field called Number of Graphics Segments Field (NUMS) is defined. Then there is a pair of LSSHn (Length of Graphics Subheader) and LSn (Length of Graphics Data) fields for each Graphics Segment defined in NUMS.

After the fields are defined for Graphics Segments, there is a special 3-byte field called NUMX. It is solely reserved for future use and its value should be equal to 000.

Following the NUMX field, Text Segments are defined in the same manner as Image and Graphics Segments. A field called NUMT defines how many text segments are in the file. Then a pair of two fields for each text segment follows called LTSHn and LTn.

Following the Text Segment definition are the fields for Data Extension Segment (DES). A NUMDES field defines how many DES segments are in the file, and then an LDSHn and LDn field are defined for the length of each DES subheader and data, respectively.

Finally, following the Data Extension Segment is the Reserved Extension Segment (RES) definition. A similar NUMRES fields defines how many RES segments are in the file. This is then followed by a pair of LRESHn and LREn to define the length of an RES subheader and data, respectively.

Simple Image with 1 Image Segment (LISH1 and LI1 present):

**Table 4-11. NITF File Header Segment Length Example**

| Field | Value | Description |
|---|---|---|
| NUMI | 001 | 1 image segment |
| LISH1 | 000439 | 439 bytes is the length of 1st Image Subheader |
| LI1 | 0002097152 | 2,097,152 bytes is the length of 1st Image Segment of data |
| NUMS | 000 | 0 Graphics Segments |
| NUMX | 000 | Reserved for Future Use |
| NUMT | 000 | 0 Text Segments |
| NUMDES | 000 | 0 Data Extension Segments |
| NUMRES | 000 | 0 Reserved Extension Segments |

Now consider changing these fields such that they do not reflect the actual content in the file. If the example in Table 4-11 is modified to show 0 image segments, parsers may not open the file since there are no segments in the file. An example is shown in Table 4-12, this results in a NITF viewer giving an error that there are no segments in the file.

**Table 4-12. Data Hiding Changing Segment Count**

| Field | Value | Description |
|---|---|---|
| NUMI | 000 | 0 image segments (but data is there later) |
| LISH1 | 000439 | 439 bytes is the length of 1st Image Subheader |
| LI1 | 0002097152 | 2,097,152 bytes is the length of 1st Image Segment of data |
| NUMS | 000 | 0 Graphics Segments |
| NUMX | 000 | Reserved for Future Use |
| NUMT | 000 | 0 Text Segments |
| NUMDES | 000 | 0 Data Extension Segments |
| NUMRES | 000 | 0 Reserved Extension Segments |

**PRODUCT: NITF 2.1**

**LOCATIONS:**

The Number of Image Segments (NUMI) Field starts at byte 360 and is 3 bytes long.

The Length of $n^{th}$ Image Subheader (LISHn) field start at Byte 363 and each field is 6 bytes long. They will alternate following after each LIn field that exists.

The Length of $n^{th}$ Image Segment (LIn) field start at Byte 369 and each field is 10 bytes long. They will alternate following after each LISHn field that exists.

The definition of Graphics Segments (NUMS, LSSHn, LSn) following the last LIn.

NUMX follows the last LSn (if it exists), and is only 3 bytes long.

The definition of Text Segments follows NUMX, which is then followed by the definition of Data Extension Segments (NUMDES/LDSHn/LDn) and Reserved Extension Segments (NUMRES/LRESHn/LREn).

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding –** If Segment Numbers are inaccurate, there could be data hiding. For example, if NUMI equals 0, and there is actual data in the file, many parsers may ignore the segment.

**1** **Validate:** Ensure that for each NUM*x* (*x* is for each segment) value, there is a corresponding segment in the file at the right location.

**2** **Validate**: Check that NUMX (Reserved for Future Use Field) is equal to 000.

**Data Attack -** Although no instances have been found, any length property may introduce risk and may cause errors with parsing software. Any length field that is not verified may introduce an attack if memory is allocated for segment data.

**3** **Validate:** Check that the NUMI, NUMS, NUMT, NUMDES, and NUMRES fields are all between 000 and 999.

**4** **Validate**: Check that for each NUMx (x is for each segment), there is an appropriate number of LxSHn and Lxn fields. This will require parsing the rest of the file to determine.

**5** **Validate**: Check for each LxSHn (x is for each segment) that the value ranges from its specified range in Table A-1 of the NITF specification.

**6** **Validate**: Check that for each Lxn (x is for each segment) that the value ranges from its specific range in Table A-1 of the NITF specfication.

**7** **Validate**: Check for each Lxn that the data in the file matches the size of data and aligns with the next segment. Note that a value of all 9s may be used in certain cases when the length was not known at the time it was created.

8 **Reject**: Reject files with lengths that are not correct. Incorrect lengths will only be discovered when later fields are parsed and determined to have incorrect values that fail validation.

## NITF.4.11:END

## NITF.4.12:NITF HEADER – USER DEFINED HEADER DATA

### DESCRIPTION:

NITF defines additions to the File Header to support including Tagged Record Extensions (TRE). This is all done through a User Defined Header near the end of the File Header. Three possible fields are introduced: User Defined Header Data Length (UDHDL), User Defined Header Overflow (UDHOFL), and User-Defined Header Data (UDHD).

User defined headers may exist in other subheaders as well, such as the image subheader. If a User-Defined field is discovered in later parts of the file, this construct applies to this as well.

The User Defined Header supports adding TREs. A TRE is defined by NITF as a "collection of data fields that provides space within the NITF file structure for adding, as yet unspecified, future capabilities to the standard" [1]. It is typically metadata regarding the image file and useful associated information. A TRE consists of three fields: Tag, Length, and Data. There are two different types of TREs: a Control Extension (CE) TRE and a Registered Extension (RE) TRE. Both have the same Tag/Length/Data format. For a Control Extension TRE, both the Tag and the Data values are supported by NITFS Technical Board (NTB), and their structure is controlled. Basically, these are known or controlled TRE blocks of data. A Registered Extension TRE is less constrictive as the Data format of the TRE is not controlled and may be in any format. All Tag values must be registered with the NTB to avoid duplication of Tag values. TRE data may reside in the UDHD portion of the NITF file header in any order, and may also overflow in a Data Extension Segment (DES), which is defined later in the file.

The UDHDL field is equal to 0 if there are no TREs defined in the UDHD. If this field is zero, there are no UDHOFL and UDHD fields present in the NITF File Header, they are conditional.

If TREs exist in the UDHD, the value of UDHDL is equal to the total length of the TRE data plus 3. The UDHOFL is 3 bytes long.

If the TRE data is too large to fit within the UDHD field, it may overflow in a Data Extension Segment (DES) defined later in the file. If this is the case, the UDHOFL indicates the sequence number of the DES where the remainder of the TRE data is located. If there is no overflow, this field is conditional and does not exist.

The TRE Data Structure is identified in a construct in Section 5.6.

### PRODUCT: NITF 2.1

#### LOCATIONS:

UDHDL immediately follows the last LREn from construct NITF.4.11. It is 5 bytes in length. The UDHOFL field is conditional and if it exists, would follow the UDHDL field and is 3 bytes long. The UDHD field follows this field and its size is the value of UDHDL minus 3, which could be on the order of tens of kilobytes.

User Defined data may also exist in image subheaders.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** User defined data that is not defined by any specification could potentially lead to a hiding risk if this information was placed here with this intention.

1 **Validate**: If UDHDL field is non-zero, check that it equals the size of the UDHD field plus 3.

2 **Validate**: If UDHOFL is implemented, check that the overflowed TRE data exists in the correct and existing DES.

**Data Disclosure -** Data may exist in TREs that might be accidently released.

3 **External Filtering Required**: Pass TRE data to an external filter.

## NITF.4.12:END

## NITF.4.13:NITF HEADER – EXTENDED HEADER DATA

#### DESCRIPTION:

The Extended Header is similar to the User-Defined Header as it also contains more Tagged Record Extensions (TRE). These TRE blocks of data are approved and under control of the Imagery Standards Management Committee (ISMC).

Extended headers may also exist for individual subheaders. If this is the case, then this construct applies to each subheader Extended Data that is located at the end of the subheader.

The Extended Header definition in the NITF File Header is almost identical to the User-Defined Header. It contains a field call Extended Header Data Length (XHDL). If this value is zero, there are no more fields in the NITF File Header. If non-zero, the value should be equal to the size of the TREs in the Extended Header Data (XHD) plus 3. There is an Extended Header Data Overflow (XHDLOFL), a conditional field, which is used to overflow the data into a Data Extension Segment (DES). If there is no overflow, this value is 0. Lastly, there is the Extended Header Data (XHD) which contains the TRE values.

### PRODUCT: NITF 2.1

#### LOCATIONS:

The XHDL field follows the UDHD field from the previous construct. It is 5 bytes long.

The XHDLOFL field, if implemented, follows XHDL, and is 3 bytes long.

The XHD field follows the XHDLOFL field, and its size is equal to XHDL minus 3.

Extended Data may also exist in image subheaders.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding:** If sizing is not correct or overflow data is not linked correctly, it may skip over data and information could be hidden.

**1** **Validate**: If XHDL field is non-zero, check that it equals the size of the XHD field plus 3.

**2** **Validate:** If XHDLOFL is implemented, check that the overflowed TRE data exists in the correct and existing DES.

**3** **Remove:** Remove the Extended Data, remove all the data until the next segment, which will require changing all the length fields.

**4** **Reject:** If the size is not correct or overflow does not line up, reject the file.

**Data Disclosure:** Data may exist in TRE data that might be accidently released.

**5** **External Filtering Required**: Pass TRE data to an external filter.

### NITF.4.13:END

# 5. NITF – NITF FILE SEGMENTS

This section covers the subheaders and data portion of each possible segment in the NITF file. This data immediately follows the last byte of the NITF File Header discussed in the previous section. Each segment contains a subheader and a block of data. There may be zero or numerous segments of each type in NITF.

There are several fields that are introduced before describing each segment and its subheader. These fields are common throughout various subheaders and are combined into single constructs listed below. One example is the segment classification field. All segments provide the same classification fields as the NITF File Header; however, the fields in each subheader apply only to that segment of data. The following construct, NITF.5.1, deals with all of the classification fields as it applies to a subheader.

Display Levels and Attachment Levels are also defined as constructs in this section since they are common to both images and graphics. They allow for the positioning and display order of segments within the NITF file.

### NITF.5.1: ALL SUBHEADERS –CLASSIFICATION FIELDS

**DESCRIPTION:**

All subheaders following the NITF File Header contain fields regarding the security classification of only the following segment of data. While the scope is different from the overall NITF File Header Security Classification fields, the structure is exactly the same. Recommendations that are listed in Section 4 for the NITF File header shall apply to the same classification fields in each subheader. They have the same name, but instead of "File", they are preceded by "Image" or "Graphic". They are the same type of field, introduce the same risks, but refer to that segment of data instead of the overall file.

The only important distinction in this construct is to compare the classification labels between the segment and overall file. For example, if the overall file classification is less than a specific segment, there is a problem.

**PRODUCT: NITF 2.1**

**LOCATIONS:**

The classification fields are in each subheader following the first few fields as defined in Appendix A of the NITF standard. Typically there are a few fields prior to the classification fields that introduce the Subheader and its type (Image, Graphic, Text, etc.).

**CONCERNS AND RECOMMENDATIONS:**

**Data Disclosure -** If classification information is different from the overall NITF File header, there could be an accidental release of information if this field is not checked since it is assumed the subheader is labeled correctly. All of the risks from previous sections on classification fields apply as well.

1 **Validate**: Check for each field in this construct that it is consistent with the NITF file header.

2 **Remove:** If classification is greater, remove the subheader and segment data. This will require a rebuild of the entire file (including the NITF File Header).

3 **Reject**: Reject files where the subheader classification details is greater than the overall NITF File header.

## NITF.5.1: END

## NITF.5.2: IMAGE AND GRAPHIC SUBHEADERS – DISPLAY LEVEL

**DESCRIPTION:**

All graphic and image data in a NITF file are layers in a composite image. Content is stacked on top of one another to provide a view of associated data. The Display Level Field in each subheader defines its visibility compared to other segments. The field is 3 bytes in length and can range from 000 to 999. A mockup example of display level is shown below with three different types of possible segment data, an image and 2 graphics.

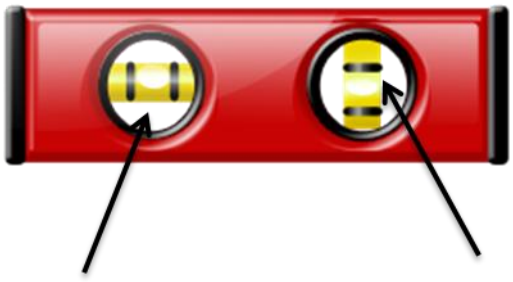

**Figure 5-1. Display Level Example**

**Base Level Image:**
Display Level: 001

**Graphic Arrow (1):**
Display Level: 002

**Graphic Arrow (2):**
Display Level: 003

The Display Level field provides a unique number for the content, the higher the value, the higher on the stack of content it resides. The standard specifies if two segments of data share a pixel location, the higher display will be shown.

### PRODUCT: NITF 2.1

**LOCATIONS:**
The Display Level can be found in the Image and Graphic Subheaders.

**CONCERNS AND RECOMMENDATIONS:**
**Data Hiding -** Changing the display level to a different value may introduce data hiding.

**1** **Validate**: Check that the Display Level is unique throughout the entire NITF file.

**2** **Replace**: Replace all the layered content by flattening it with a single image in a single segment; this will break the original functionality.

## NITF.5.2: END

## NITF.5.3: IMAGE, GRAPHIC, AND TEXT SUBHEADERS – ATTACHMENT LEVEL

**DESCRIPTION:**
The Attachment Level Field associates different segments of data which allows for them to be combined into groups. This is helpful if content is deleted, moved, or rotated, as all segments with the same attachment level will be impacted by the same operation. An Attachment Level must be equal to a previous Display Level Number of another segment (i.e., they must have the same numerical value). Attachment Levels create a hierarchy of objects by relating them to a Display Level. If content of a Display Level is moved, any object with the Attachment Level equal to that Display Level will also be moved. These objects are considered to be children objects to the matching Display Level number (i.e., the parent object). The Text Subheader defines only an attachment level (no display level) as text can only be attached to the other existing content.

The minimum display level of content shall have an attachment level of zero, which means "unattached." This field is 3 bytes in length and can contain values from 000 to 998. A valid of 999 is not allowed since it must match up to an existing Display Level, the largest Display Level it can be attached to is 998.

### PRODUCT: NITF 2.1

**LOCATIONS:**
Attachment level fields are found in the Image and Graphic Subheaders.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding -** Content with Attachment Levels not equal to a known Display Level may introduce unknown behavior in the application. An application may not know how to attach a segment to an existing layer, which might misplace information.

**1** **Validate**: Check that if the Attachment Level is greater than zero, then a Display Level field of the same number is also present within the file.

**NITF.5.3: END**

# 5.1 Image Segments

## NITF.5.4: IMAGE SUBHEADER – INTRODUCTION

### DESCRIPTION:

The Image Subheader is present for each Image Segment in the file. The subheader has many fields, and this section only covers the first 5 fields. The remaining fields are in following construct sections.

Image subheaders contain a field called File Part Type which shall be equal to "IM". All subheaders contain a File Part Type field as a unique identifier to the type of segment. The rest of the fields in this construct are described below.

**Table 5-1. Image Subheader Fields**

| Field | Description |
|---|---|
| File Part Type | 2 characters: shall be "IM" for Image segments. |
| Image Identifier 1 (IID1) | 10 characters |
| Image Date and Time (IDATIM) | 14 digits |
| Target Identifier (TGTID) | 17 characters in the format of BBBBBBBBBBOOOOOCC as specified in FIPS PUB 10-4 |
| Image Identifier 2 (IID2) | 80 characters of any type of information |

Example from NITF Standard:

**Table 5-2. NITF Image Subheader Example**

| Field | Value |
|---|---|
| File Part Type | IM |
| Image Identifier 1 | 0000000001 |

| | |
|---|---|
| Image Date and Time | 19960825203147 |
| Target Identifier | |
| Image Identifier 2 | 1996238CY0212345678ABCD25AUG19952031F |

### PRODUCT: NITF 2.1

#### LOCATIONS:

These fields begin the Image Subheader which follows the NITF File Header.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** If any of the fields are not parsed, data could be hidden at this location.

**1** **Validate**: Check that the File Part Type is IM, and that its byte location in the file lines up with length information from NITF File Header.

**2** **Validate**: Check that Image Date and Time is in the correct format.

**3** **Validate**: Check that Target Identifier is correct according to FIPS PUB 10-4

**Data Disclosure -** There are several fields in this header that contain metadata which might accidently release information.

**4** **External Filtering Required**: Pass Image Identifier 1 and 2 fields to an external filter.

## NITF.5.4: END

## NITF.5.5: IMAGE SUBHEADER – IMAGE SOURCE, SIZE, AND LOCATION

### DESCRIPTION:

Following the Image Classification fields, there are 10 fields (one is conditional) that provide additional information about the data in the image segment. This includes the source, the size (rows/columns) and other image categories and location information. The following table lists these fields and a description about each one. Table A-3 in the NITF standard introduces these fields, their lengths, and possible list of values.

### Table 5-3. Image Subheader Fields

| Field | Size (bytes) | Description |
|---|---|---|
| Image Source | 42 | A field of free text defining the source of the image. |
| Number of Significant Rows in Image | 8 | Number of rows in image that contain actual image data. (Index=0) |
| Number of Significant Columns in Image | 8 | Number of columns in image that contain actual image data. (Index=0) |
| Pixel Value Type | 3 | 3 characters used to define the data type of each pixel. Possible values are “INT”, B”, “SI”, “R”, and “C”. |
| Image Representation | 8 | Field to describe image for processing. Valid values are “MONO”, “RGB”, “RGB/LUT”, “MULTI”, “NODISPLY”, “NVECTOR”, “POLAR”, “VPH”, and “YCbCr601”. |
| Image Category | 8 | Field to describe the image, or the category of the image. This ranges from raster map data to XRAY or CAT scan data. Table A-3 in the NITF standard lists the value ranges for this field. |
| Actual Bits-Per-Pixel-Per-Band | 2 | This value represents the number of significant bits without compression for each band of each pixel [1]. Valid values are 01 to 96. |
| Pixel Justification | 1 | Left or Right Justified for Significant Bits, values are L or R. |
| Image Coordinate Representation | 1 | Contains codes for coordinate representation, for the location of the image. Valid values are U, G, N, S, D, or Space (0x20). |
| Image Geographic Location (Conditional) | 60 | If the Image Coordinate Representation is Space (0x20), this field does not exist. If it does exist, it contains an approximate geographic location. |

**PRODUCT: NITF 2.1**

**LOCATIONS:**
These fields follow the Image Classification fields in the Image Subheader.

**CONCERNS AND RECOMMENDATIONS:**
**Data Hiding -** Changing the image parameters may have an impact on parsing applications and hide information.

**1** **Validate**: Check that Pixel Value Type, Image Representation, and Image Category contain values from their list defined in Table A-3 of the NITF Standard.

**2** **Validate:** If possible, check that the number of significant rows/columns matches up to the image data.

**3** **Validate:** If possible, check that the Actual Bits-Per-Pixel-Per-Band matches with the image data.

**Data Hiding and Data Disclosure -** Any free text field in this part of the subheader may introduce sensitive information.

**4** **External Filtering Required**: Pass Image Source and Image Geographic Location (if exists) to an external filter.

**NITF.5.5: END**

## NITF.5.6: IMAGE SUBHEADER – IMAGE COMMENTS

### DESCRIPTION:

The Image Subheader supports up to 9 different comment fields. They are used for free text of information about the image segment. The first field is the Number of Image Comments field. This is a required field that can contain any value between 0 and 9. This field defines how many comment fields follow. If zero, there are no following comment fields that exist in the Image Subheader.

An example from the NITF standard is below:

**Table 5-4. Image Subheader – Comments Example**

| Field | Size (bytes) | Value | Description |
|---|---|---|---|
| Number of Image Comments (NICOM) | 1 | 3 | Defines how many fields follow for image comments, Minimum is zero, maximum is 9 |
| Image Comment 1 (ICOM1) | 80 | This is a comment on Major Test Facility base and associated inset. This file w | First block of comments. |
| Image Comment 2 (ICOM2) | 80 | as developed at Fort Huachuca, Arizona. It shows the Joint Interoperability Tes | Second block of comments. |
| Image Comment 3 (ICOM3) | 80 | t Command Building and associated range areas. | Third block of comments. |

### PRODUCT: NITF 2.1

### LOCATIONS:

The Number of Image Comments field follows the previous fields defined in NITF.5.3. Depending on the value in this field, there are 0-9 Image comment fields, each of 80 characters.

### CONCERNS AND RECOMMENDATIONS:

**Data Hiding, Data Attack, and Data Disclosure -** Sensitive data, executable code, or user supplied hidden data may exist in these comment fields.

**1** **Validate:** Check that each comment has only alphanumeric characters.

**2** **Remove:** Remove all image comment fields and set the Number of Image Comments field to 0. This will require rebuilding and realigning the file and lengths throughout.

**3** **Replace**: Replace all existing image comment fields with spaces to avoid changing lengths of the file elsewhere.

**4** **External Filtering Required**: Pass all image comment fields to an external filter.

## NITF.5.6: END

## NITF.5.7: IMAGE SUBHEADER – IMAGE COMPRESSION

### DESCRIPTION:

The Image Subheader contains fields for defining the mode of compression used for the data in the data segment following the Image Subheader. This is an important field because it identifies the image type of the data following the Image Subheader. Two fields are the focus of this section: one called Image Compression (Labeled as IC) and the other is Compression Rate Code (Labeled as COMRAT). Compression Rate code is conditional, only if compression is used. The Image Compression field defines if an image mask is used. If an image mask is specified, then the Image Mask Table will be present after the end of the Image Subheader. The Image Compression field defines several possibilities in Table A-3 of the NITF Standard, and a list of them in also shown below in Table 5-5.

### Table 5-5. Image Compression Types

| Image Compression Value | Image Type | Compression Reference |
|---|---|---|
| NC/NM | No Compression/No Compression with Image Mask. | N/A |
| C1/M1 | Bi-Level/Bi-Level with Image Mask. | ITU-T T.4, AMD2 |
| C3/M3 | JPEG/JPEG with Image Mask. | MIL-STD-188-198A |
| C4/M4 | Vector Quantization/Vector Quantization with Image Mask. | MIL-STD-188-199 |
| C5/M5 | Bandwidth Compression/ Bandwidth Compression with Image Mask | NGA N0106-97 |
| C6/M6 | For future correlated multicomponent compression algorithm | N/A |
| C7/M7 | Reserved values for future complex SAR compression. | N/A |
| C8/M8 | JPEG 2000/JPEG 2000 with Image Mask. | ISO/IEC 15444-1:2000 (with amendments 1 and 2). |
| I1/M1 | Down sampled JPEG/Down sampled JPEG with Image Mask. | NGA N0106-97 |

### PRODUCT: NITF 2.1

#### LOCATIONS:
These two fields follow the last Image Comment field in the Image Subheader.

#### CONCERNS AND RECOMMENDATIONS:
**Data Attack -** These fields influence how the data segment is parsed. Incorrect compression fields or invalid compression fields will lead to incorrect image parsing in applications.

**1** **Validate**: Check that the Image Compression field is on the list of allowed fields in Table A-3 of NITF standard.

**2** **Validate**: If a mask is specified in the Image Compression field, check that the Mask table does exist following the Image Subheader.

**3** **Validate**: If a compression field indicates that compression is used, check the Compression Rate Code is a valid entry according to the Table A-3 of the NITF Standard.

## NITF.5.7: END

## NITF.5.8: IMAGE SUBHEADER – NUMBER OF IMAGE DATA BANDS

#### DESCRIPTION:
Images in NITF files can contain numerous bands of data. Most images may contain three bands, for red, green, and blue. Other imagery, such as satellite imagery, may contain multiple bands ranging from ultraviolet, visible spectrum bands, and infrared bands. This part of the image subheader defines this information about the image that follows.

The first field is called NBANDS, which define the number of Data Bands. It is related to the Image Representation field described in NITF.5.5. For example, if the Image Representation (IREP) field is RGB, then the value of NBANDS is equal to 3. This field ranges from 0 to 9 for its values, as the size of the field is a single byte. Different data band numbers relate to different image types that are discussed in the NITF standard.

If the value of NBANDS is equal to 0, a conditional field is present next called XBANDS, which represents the Number of Multispectral Bands. Valid values range from 10 to 99999.

### PRODUCT: NITF 2.1

#### LOCATIONS:
NBANDS and XBANDS follow the Image Compression fields in the Image Subheader discussed in NITF.5.7.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack -** These fields provide information for parsing the remainder of the subheader, and rely on these fields for allocating memory. Any field that represent length and then uses that create a variable block of memory introduces a potential attack risk.

**1** **Validate**: Check that NBANDS is present and contains a value from 0-9. If possible, check this value against the value in Image Representation in NITF.5.5 to verify it is correct.

**2** **Validate**: If NBANDS equals 1-9, if possible check that XBANDS is not present.

**3** **Validate**: If NBANDS equals 0, check that XBANDS exists and is a positive integer between 10 and 99999.

**4** **Reject**: Reject NITF file that contain an invalid number or inconsistent number of data bands.

## NITF.5.8: END

## NITF.5.9: IMAGE SUBHEADER – DATA BAND REPRESENTATION AND CATEGORY

#### DESCRIPTION:

For every NBANDS or XBANDS number of data bands defined in this image segment, there are a number of fields defining the "n$^{th}$ Band Representation." This involves the IREPBANDn field or Image Representation for Band *N*.

The IREPBANDn field must align with the Image Representation Field in NITF.5.5. (IREP). Valid values are LU (LookUp Table), R (Red), G (Green), B (Blue), M (Monochrome), Y (Luminance), Cb (Chrominance - blue), Cr (Chrominance - red), and it may include others that are registered. Chrominance is used to describe color information, while the Luminance describes the brightness of the image in color spaces such as YCbCr. A Look-Up Table (LUT) is used for special translation of pixel (indexes) and the values in the look-up table.

The following table below defines how the IREP field relates to possible IREBANDn values.

### Table 5-6. Image Band Representation Values

| IREP Field Value | Valid IREPBAND*n* Values |
|---|---|
| MULTI | M, R, G, B, LU, or all spaces (0x20) |
| MONO | M, LU, or all spaces (0x20) |
| RGB | R, G, B |
| RGB/LUT | LU |

| | |
|---|---|
| YCbCr601 | Y, Cb, and Cr |
| NODISPLY | All Spaces (0x20) |
| NVECTOR | All Spaces (0x20) |
| POLAR | All Spaces (0x20) |

The second field for Data Band Representation is the Image Subcategory for the Band N, or ISUBCATn. This relates directly to the Image Category Field (ICAT) defined in previous constructs. Table A-3 in the Appendix of the NITF standard lists the constraints for acceptable values in this field. Like IREPBANDn, ISUBCATn relates directly to the ICAT field, so there is a whitelist of acceptable values for this field.

There are two more fields for each Data Band called IFCn ($n^{th}$ Band Image Filter Condition), which is always set to "N" and the IMFLTn ($n^{th}$ Band Standard Image Filter Code), which is reserved and equal to all spaces.

### PRODUCT: NITF 2.1

#### LOCATIONS:

These fields in this section follow the NBANDS and XBANDS (only if it exists) fields.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding –** These fields can be validated easily since there is a list from the standard. With basic NITF files, these fields can be supplied with any value and there is no impact opening the file, thus it presents a few bytes of data hiding.

**1** **Validate**: Check that for each IREPBANDn that a valid value exists per the IREP value from the table above.

**2** **Validate:** Check that for each ISUBCATn that a valid value exists as defined per the ICAT value in Table A-3 of the NITF standard.

**3** **Replace**: If possible, replace these fields with default values, which for most is all spaces (0x20).

## NITF.5.9: END

## NITF.5.10:IMAGE SUBHEADER – IMAGE BAND LOOK-UP TABLE

### DESCRIPTION:

Following the representation and category fields for each data band are fields related to a Look-up Table, if it exists. The first field to define this is called Number of Look-Up Tables for the $n^{th}$ Image Band, or NLUTSn. This field ranges from 0 to 4 and defines how many tables exist for this image band. If this field is between 1 and 4, then a second conditional field follows called Number of LUT Entries for the $n^{th}$ Image Band. (NELUTn). It defines how large the Look-up Tables are, ranging from 1 to 65536.

Following the NETLUTn field is a table of values called the $n^{th}$ Image Band, $m^{th}$ LUT. The field consists of unsigned binary integer values, with the length equal to the NELUTn field defining the number of LUT entries for this image band.

The three fields discussed in this section are repeated for every image band in the file.

Some of these fields are conditional in this section, if conditional fields exist for some reason but they should not, they will introduce parsing errors and later fields will fail validation.

### CONCERNS:
### PRODUCT: NITF 2.1

#### LOCATIONS:

The NLUTSn and NELUTn fields appear after the ISUBCATn field for a particular image band. The Table LUTDnm (nth Image Band, mth LUT) follows the NELUTn field.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack and Data Hiding** – A look-up table could be replaced with malicious content or be used to hide existing content within an image.

1 **Validate**: If possible, determine that the Look-up Table does not contain free text or executable code or some type of signature matching.

2 **Remove:** Remove the Look-up Table and convert the Data Band to another value that does not rely on the Look-up Table. This will have an impact on the image.

## NITF.5.10: END

## NITF.5.11:IMAGE SUBHEADER – IMAGE MODE

### DESCRIPTION:

Image mode is a field within the NITF Image Subheader that defines how image pixels are stored in the image data segment. There are only 4 possible values allowed for this single byte field and they are as follows. These values relate to the compression field in the image subheader.

### Table 5-7. Image Mode Values

| Value | Definition |
|---|---|
| B | Band Interleaved by Block |
| P | Band Interleaved by Pixel |
| R | Band Interleaved by Row |
| S | Band Sequential |

If the image is JPEG compressed (Compression is C3 or M3), it must use either B, P, or S.

If Image Compression is listed as C8 (JPEG2000), M8 (JPEG2000 with Image Mask), or I1 (Downsampled JPEG), this field shall be equal to B.

If Image Compression is listed as C1 (Bi-Level) or M1 (Bi-Level with Image Mask), then this field shall be equal to B.

Vector Quantization compression (C4 or M4) means that this field shall be equal to B.

#### PRODUCT: NITF 2.1

#### LOCATIONS:

This field follows the Image Sync Code field in the Image Subheader, a reserved single byte field following the Look-up table information.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack -** Changing how the image is read may lead to unknown behavior in the application.

**1** **Validate**: Check that the value is one of the four possible options.

**2** **Validate:** If possible, check this value against the compression field.

**3** **Remove:** Remove entire image segments with invalid image modes.

**4** **Reject:** Reject NITF files with invalid image modes.

## NITF.5.11: END

## NITF.5.12:IMAGE SUBHEADER – IMAGE BLOCKS AND PIXELS

### DESCRIPTION:

The Image subheader defines the number of blocks per row (NBPR) and the number of blocks per column (NBPC) in the image, needed for parsing the image segment data. For example, if the image compression indicates JPEG and the blocks per row and column field are both 4, this would imply that the image segment data contains 4 rows and 4 columns of JPEG data, or 16 individual JPEG files. If there is an Image Mask present, a block may be absent since it contains no pixel data, but NITF viewers must be aware and must produce the empty block when rearranging the image blocks.

The image subheader also indicates the number of pixels per block horizontal (NPPBH) and pixels per block vertical (NPPBV). The total size of the entire image is multiplied by the number of blocks in each dimension. Since the dimensions (in pixels) of each block is specified here, each block has the same dimension.

There is a final field called Number of Bits Per Pixel Per Band (NBPP). There could be multiple bands defined in the NBANDS or XBANDS field defined in NITF.5.9.

### PRODUCT: NITF 2.1

#### LOCATIONS:

These five fields identified in this section follow the Image Mode field in the Image Subheader.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack -** False values in these fields may cause errors parsing the image data.

**Data Hiding –** An invalid number of blocks will only show a subset of the image, which could lead to a hidden data risk.

**1** **Validate**: Check that the number of blocks per row and column align with the image data.

**2** **Validate**: Check that the Number of Bits per Pixel per Band and per block matches what exists in the image data and matches the requirements from Table A-3 of the Appendix according to the image compression of the image data segment.

3 **Remove:** Remove image segments and their subheader if they contain an invalid entry in any of these fields.

## NITF.5.12: END

## NITF.5.13: IMAGE SUBHEADER – IMAGE LOCATION AND MAGNIFICATION

### DESCRIPTION:

The image location (ILOC) field identifies the location of the first pixel of the first line in the image for this particular image segment. If the image is attached to another segment, then this value is the offset from that origin of that segment. An unattached image may have values of all zero's indicating no offset. The image location field is in the format of RRRRCCCCC for rows and columns offset. Positive values means down and to the right, while negative values mean up and to the left.

The image subheader also provides an Image Magnification field (IMAG). This field defines the zoom magnification or reduction factor of the image. This value is relative to the original image without any magnification. Values greater than 1 will magnify the image, and values less than 1 will reduce the image (zoom out). For magnification values greater than 1, the field may contain the value "2.3". For reduction values the field may contain "/2". This means a value of 0.5. All values must be positive in this field.

### PRODUCT: NITF 2.1

#### LOCATIONS:

The ILOC field and IMAG field follow the Image Attachment Level (IALVL) field in the Image Subheader.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Changing the image location or magnification may hide contents of the image.

1 **Validate**: Ensure that the image location field and the size of the image data does not place it out of visibility.

2 **Validate**: If the image is attached to another segment, ensure the offset location does not place it out of visibility.

3 **Replace**: Replace unattached images with an image location of all zeros.

4 **Replace**: Replace magnification field with a value of 1.0 for no magnification.

## NITF.5.13: END

### 5.1.1 Image Data Masks

The Image Compression (IC) field in the Image Subheader supports modes where a block mask or a pad pixel mask can be applied to that image. To supports this, an image mask is defined immediately following the Image Subheader (immediately before the start of the Image Segment data). This section and single construct identify the fields within an Image Mask table.

**NITF.5.14: IMAGE DATA MASK TABLE**

**DESCRIPTION:**

If the compression field in the Image Subheader defines an image mask is to be used, the following fields are defined after the end of the Image Subheader. There are two types of image masks: a block mask and a pad pixel mask. The block mask is a structure that identifies blocks of the image that contain no information or data. These blocks are not transmitted as part of the file. This allows for blocks of image data to be transmitted without needed to transmit "empty" blocks of information. Consider dividing an image into numerous blocks of data as shown in Figure 5-2. There could possibly be blocks of no information that does not need to be transmitted with the image, such as Block(1,1). If the empty block is not transmitted as part of the whole image, the image could not be properly restructured. The Block Mask will identify how to restructure the image and which blocks are empty. If this were a NITF file, there would be 8 blocks or images located in the image segment data, the mask table would identify the first block as empty.

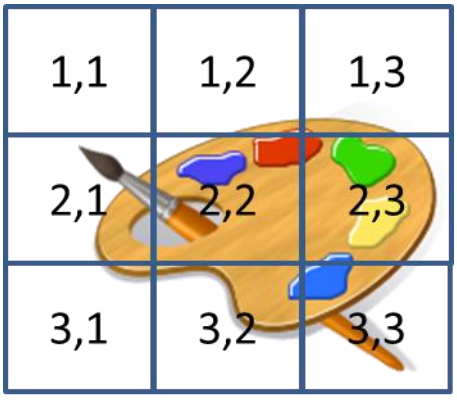


**Figure 5-2. Blocked Image with Empty Block**

The Pad Pixel Mask is used to fill an image block to the nearest block boundary. Consider a blocked image again shown in Figure 5-2. Every other block will contain an area with no pixel values, blocks (1,2), (1,3), and (3,1) contain the most pixels with no value. Instead of transmitting the full block with many "empty" pixels, a pad pixel mask can be used to describe that information.

**Table 5-8. Image Mask Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| Blocked Image Data Offset (IMDATOFF) | 4 | This field is used to determine the offset into the image where the mask is to be applied. |
| Block Mask Record Length (BMRLNTH) | 2 | This field is used to define the length of each Block Mask Record (BMRnBNDm from below). |
| Pad Pixel Mask Record Length (TMRLNTH) | 2 | This field is used to define the length of each Pad Pixel Mask Record (TMRnBNDm from below). |
| Pad Output Pixel Code Length (TPXCDLNTH) | 2 | This field identifies the length of the next field, the Pad Output Pixel Code. If this field is zero, the next field does not exist. |
| Pad Output Pixel Code (TPXCD) | The value defined the field Pad Output Pixel Code Length TPXCDLNTH | This field allows a parser to identify pad pixels. This should be a unique value in the image and it is used to represent a pad value. |
| Block n, Band m Offset (BMRnBNDm) | 4 | This field repeats for each block mask. This field repeats, there is one 4 byte field per each block. |
| Pad Pixel n, Band m (TMRnBNDm) | 4 | This field repeats for each pixel mask (4 bytes per block of data). |

In this example shown in Figure 5-3, the TMRnBNDm field was modified to inject a small sentence; the length of the field was not changed. The IMDATOFF field was replaced with a 4 character word (“This”) and the TMRnBNDm field was replaced with “There are hidden bytes here .” Some applications were able to recover and display the image properly despite this modification.

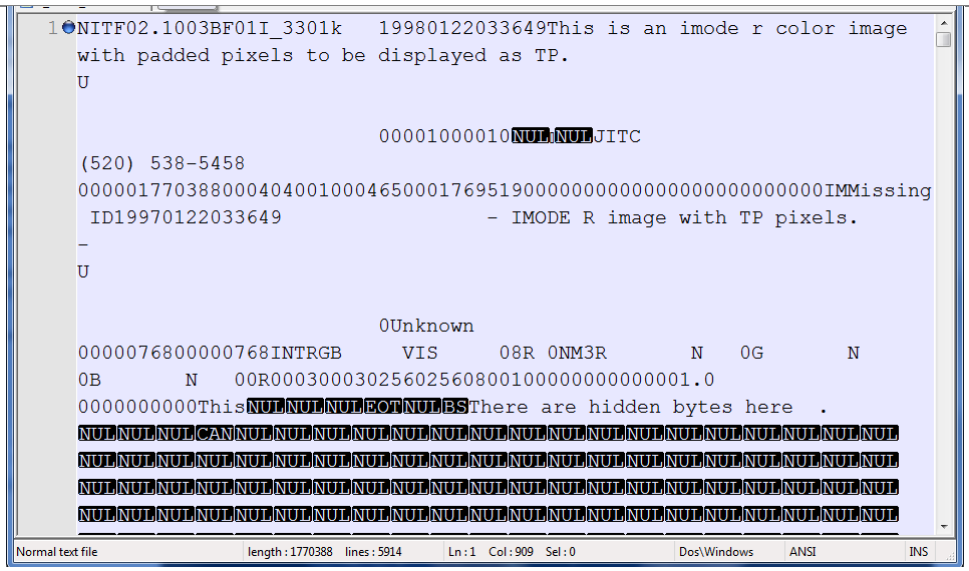


**Figure 5-3. NITF with modified Image Mask**

### PRODUCT: NITF 2.1

#### LOCATIONS:

The fields defined in this construct are at the beginning of the Image Segment data immediately following the Image Subheader.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding –** The Image Mask could be used to alter the image and provide information that might not be accurate with the image data. Since it ultimately affects the image, it could possibly be used to hide information.

1 **Validate:** Check that the IMDATOFF field is valid and points to a valid offset.

2 **Validate**: Check that the data of the Image Mask size lines up to the BMRLNTH or TMRLNTH field and the size of the BMRnBNDm/TMRnBNDm fields.

3 **Remove:** Remove the entire Mask Table, which will require reconstructing the image data to reflect the whole image with empty pixels or blocks.

## NITF.5.14: END

## 5.2 Graphic Segments

### NITF.5.15:GRAPHIC SUBHEADER – INTRODUCTION

**DESCRIPTION:**

The Graphic Subheader is located following all the image segments located in the file. Graphics in NITF files are currently stored as a Computer Graphics Metafiles (CGM). It is the only supported type for graphics according to the standard. This is different from some of the image format types defined in the NITF.5.7 Construct on Image Compression. CGM is defined by MIL-STD-2301A. The first three fields in the Graphic Subheader contain identifiers and names that describe the graphic. This section covers only the fields listed below.

**Table 5-9. Graphic Subheader Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| File Part Type (SY) | 2 | Shall be SY |
| Graphic Identifier | 10 | Alphanumeric identification code, assigned by application. |
| Graphic Name | 20 | Alphanumeric string for the name of the graphic. |

**PRODUCT: NITF 2.1**

**LOCATIONS:**

The Graphic Subheader follows the final image segment in the file. It appears for each graphic segment in the file.

**CONCERNS AND RECOMMENDATIONS:**

**Data Attack -** Invalid lengths of data may lead to unknown behavior in an application.

**1** **Validate**: Check that the File Part Type is SY and that its location lines up with the NITF File Header and the lengths of the segment.

**Data Hiding and Data Disclosure -** Sensitive data or hidden data may exist in the Graphic Identifier and Name fields.

**2** **Replace**: Replace Graphic Name with its default value of all spaces (0x20).

**3** **External Filtering Required**: Pass the Graphic identifier field to an external filter.

**4** **External Filtering Required**: Pass the Graphic Name field to an external filter.

### NITF.5.15: END

## NITF.5.16:GRAPHIC SUBHEADER – LOCATION AND COLOR

### DESCRIPTION:

The Graphic Subheader provides fields for defining the location of the graphic, upper left corner of the graphic, as well as the color.

The Graphic Location field (SLOC) provides the position of the graphic relative to the Image or Graphic Location field where it is attached. If unattached, the position is relative to the origin. The field is defined with 10 bytes, in the format of RRRRRCCCCC (Rows and Columns). This field can specify negative rows and columns, along with positive values. Positive values indicate offsets down and right, while negative values indicate up and left. Positive values do not include a "+" sign as one of the digits; however, a negative value would reserve the most significant byte as "-".

For location, there are two more fields called First and Second Graphic Bound Location. The First location contains a field in the format of (rrrrrccccc); same position as the Location field, but defines the upper left bounding box of the graphic. The Second Graphic Bound Location identifies the lower right. All position values are relative to the Location Field of the attached object. If unattached, position is relative to the origin. Bounding locations are useful to determine if there is overlap with other content and to ensure its visibility with the rest of the segments.

The Graphic Color is defined through two different fields: the first one is called Graphic Type. At the moment the only valid value is C for Computer Graphic Metafile (CGM). The second field is called Graphic Color which shall be equal to C if color is involved with the graphic or M if monochrome.

### PRODUCT: NITF 2.1

#### LOCATIONS:

Most of the fields discussed in this section follow the Attachment Level field for Graphics in the Graphic Subheader. The Graphic Type field is the only field discussed here located before the Display Level and Attachment Level fields for this graphic.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Changing location fields and bound locations to overlap with other content may introduce data hiding.

1 **Validate**: Check that Graphic Location field does not move content out of visibility. The coordinates of the object will need to be calculated relative to the entire image.

2 **Validate**: Check that Bound Locations place the Graphic within visibility and not entirely overlapping other content.

## NITF.5.16: END

## 5.3 Text Segments

### NITF.5.17:TEXT SUBHEADER – INTRODUCTION

#### DESCRIPTION:

The first few fields in the Text Subheader introduce metadata regarding the block of text. The following table defines the first few fields and a description of each.

**Table 5-10. Text Subheader Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| File Part Type (TE) | 2 | Shall be "TE" for Text. |
| Text Identifier (TEXTID) | 7 | A user-defined/application defined identification code |
| Text Attachment Level | 3 | Covered in Construct NITF.5.3. |
| Text Date and Time | 14 | Timestamp for origination of the text. |
| Text Title (TXTITL) | 80 | Contains the free-text title of the block of text. |

#### PRODUCT: NITF 2.1

#### LOCATIONS:

The Text Subheader follows the final Graphic Segment data in the file.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Values in the fields could be substituted with hidden data.

**1** **Validate**: Check that the File Part Type is TE.

**2** **Validate**: Check that the Text Date and Time is a valid timestamp.

**3** **Replace**: Replace the Text Identifier field with all zeros.

**4** **Replace**: Replace the Text Title with all spaces (0x20).

**Data Hiding and Data Disclosure -** Free-text information about the title of the text could accidently disclose information or contain any hidden data.

**5** **External Filtering Required**: Pass the contents of Text Title to an external filter.

### NITF.5.17: END

### NITF.5.18:TEXT SUBHEADER – TEXT FORMAT

### DESCRIPTION:

The format of the text is defined by a single field called Text Format (TXTFMT) in the Text Subheader. The only possible values are MTF (United States Message Text Format), STA (Standard - BCS), UT1, Extended Character Set (ECS), and U8S (UTF-8 Subset). This value will indicate the format of the following text segment. Validation of the text segment data should be done independent of this field's value. Invalid field values might force some viewers to ignore or improperly parse the data in the text segment, so this field shall be validated for correctness.

### PRODUCT: NITF 2.1

#### LOCATIONS:

The Text Format field follows the Encryption field and Classification fields in the Text Subheader.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Invalid field values might force some viewers to ignore or improperly parse the text that follows.

**1** **Validate**: Check that the Text Format field contains a valid value from the standard.

**2** **Validate**: Check that the actual Text following the subheader contains characters from the correct text format.

**3** **Remove**: Remove Text Segments (Subheader and Data) with an invalid Text Format Field. This will require realigning the file and changing lengths throughout.

## NITF.5.18: END

## NITF.5.19:TEXT SUBHEADER – EXTENDED DATA

### DESCRIPTION:

The Text Subheader supports three additional fields that may contain TRE data, similar to the NITF File header and how it incorporates Extended Data. The three fields are described in the table below.

**Table 5-11. Text Subheader Extended Data Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| Text Extended Subheader Data Length (TXSHDL) | 5 | Required Field that contains the length of the next two fields.<br><br>If this field is zero, the other two fields do not exist. |

| | | |
|---|---|---|
| Text Extended Subheader Overflow (TXSOFL) | 3 | Equals 0 if TRE data in the next field does not overflow.<br><br>If non-zero, contains the sequence number of the DES that contains the overflowed data. |
| Text Extended Subheader Data (TXSHD) | Equal to (TXSOFL – 3) | Contains TRE data approved by ISMC. TRE data is only for the text in the segment. |

**PRODUCT: NITF 2.1**

**LOCATIONS:**
These three fields are defined in each Text Subheader (TXSHD and TXSOFL are Conditional). They follow the Text Format field discussed previously.

**CONCERNS AND RECOMMENDATIONS:**
**Data Attack -** Invalid lengths may introduce unknown behavior in applications.

1 **Validate:** Check that the length of the Subheader data is accurate with the rest of the Text Subheader and Segment.

**Data Hiding and Disclosure -** TRE data may contain hidden or sensitive content.

2 **Remove:** Remove all TRE data from this Text Segment, set the TXSHDL field to 0. This will require realigning the file and changing lengths throughout.

3 **Replace:** Replace the TRE data with all zeros.

4 **External Filtering Required:** Pass all TRE data to an external filter.

**NITF.5.19: END**

## 5.4 Data Extension Segments

There are three different types of subheaders for a Data Extension Segment (DES). A normal DES Subheader is discussed in the first section. A DES can serve as an overflow for TRE data from the NITF File Header. This subheader structure is slightly different and covered in 5.4.2. The final DES Subheader is for a Streaming File Header. This is the case when the length is not known ahead of time, which has been discussed in the NITF File Header section. The Stream File Header for a DES is described in 5.4.3.

## 5.4.1 Traditional Data Extension Segments

### NITF.5.20: DES SUBHEADER – INTRODUCTION

**DESCRIPTION:**
The Data Extension Segment (DES) contains only a few fields to introduce the data segment. Since this segment is additional data, the subheader is one of the smaller subheaders. The first three fields are covered in this section, as the remainder of the DES subheader is covered earlier in Section 5. The following table identifies the first three fields with a description.

**Table 5-12. Data Extension Subheader Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| File Part Type (DE) | 2 | Shall be "DE" |
| Unique DES Type Identifier (DESID) | 25 | ISMC Registered values for an identifier. |
| Version of the Data Definition (DESVER) | 2 | Version number of the DES, which is also registered with the identifier |

**PRODUCT: NITF 2.1**

**LOCATIONS:**
These three fields following the last remaining Text Segment.

**CONCERNS AND RECOMMENDATIONS:**
**Data Hiding -** Invalid values here may introduce hidden data.

1 **Validate**: Check that the File Part Type is DE.

2 **Validate**: Check that the Type Identifier and Version belong to a list of registered values by ISMC.

### NITF.5.20: END

## 5.4.2 Overflow Data Extension Segments

### NITF.5.21: DES OVERFLOW SUBHEADER

**DESCRIPTION:**
The Data Extension Segment (DES) contains only a few fields to introduce the data segment. Since this segment is additional data, the subheader is one of the smaller subheaders. The first three fields are covered in this section, as the remainder of the DES subheader is covered earlier in Section 5. The following table identifies the first three fields with a description.

## Table 5-13. Overflow Data Extension Subheader Fields (i)

| Field | Size (bytes) | Description |
|---|---|---|
| File Part Type (DE) | 2 | Shall be "DE" |
| Unique DES Type Identifier (DESID) | 25 | Shall be "TRE_OVERFLOW" |
| Version of the Data Definition (DESVER) | 2 | A registered version number. |

Following the first three fields are the Classification fields common to all subheaders. This has been addressed in earlier constructs. After the Classification fields, there are 4 more different fields shown below in the following table.

## Table 5-14. Overflow Data Extension Subheader Fields (ii)

| Field | Size (bytes) | Description |
|---|---|---|
| Overflowed Header Type (DESOFLW) | 6 | This implies that a TRE would not fit into the NITF File header or other subheader. Its value is where the original TRE data was defined: UDHD, UDID, XHD, IXSDH, SXSHD, or TXSHD. |
| Data Item Overflowed (DESITEM) | 3 | Present if DESOFLW field is present |
| Length of DES-Defined Subheader Fields (DESSHL) | 4 | The length of the next field. |
| DES-Defined Data field (DESDATA) | Defined in DESSHL | Binary or ASCII character data formatted in user-defined TRE. |

### PRODUCT: NITF 2.1

#### LOCATIONS:

The DES Subheader location is defined by the NITF File header (through byte length calculations), as it follows the last Text Segment in the NITF File.

#### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Invalid values here may introduce hidden data as the application may have unknown behavior.

**1** **Validate**: Check that the File Part Type is DE.

**2** **Validate**: Check that the Type Identifier is TRE_OVERFLOW.

**3** **Validate**: Check that the Overflowed Header Type is among one of the valid options from the table above.

**Data Attack and Data Hiding -** User-defined data in binary or ASCII may be a place for executable code or hidden data.

**4** **Remove:** Remove user-defined TRE and the original TRE data that has overflowed.

**Data Disclosure -** TRE data may contain user-defined alphanumeric which might accidently disclose information.

**5** **External Filtering Required:** Pass all TRE data to an external filter.

**NITF.5.21: END**

## 5.4.3 Streaming Header Data Extension Segments

The NITF File Header supports creating files that have unknown lengths at the time of their creation. This is used for time critical operations, where the initial NITF header and classification fields are filled into the header but not the length fields. A special DES segment is used for the streaming data, which is preceded by a special DES Subheader. In the DES Subheader for that streaming data is the replacement header. The replacement header should replace the entire NITF File Header that was originally in the file (with unknown lengths). Since the new header is available, systems can retransmit a newly built NITF file with all lengths known prior to constructing the file (a new file built without the Streaming DES data).

**NITF.5.22:STREAMING FILE HEADER – DATA EXTENSION SEGMENT**

**DESCRIPTION:**

Data Extension Segment (DES) Subheaders may also utilize a Streaming File Header, where the size of the data is unknown at the time of file creation. File lengths and header lengths can be all "9" (all bytes equal to the value '9')which indicate a streaming file of unknown size. Additional fields in the DES Subheader are defined in this section which allow for a streaming file. The first three fields of the DES Streaming Subheader are defined below.

**Table 5-15. Data Extension Segment Subheader Fields for Streaming Header (i)**

| Field | Size | Description |
|---|---|---|
| Data Extension Subheader (DE) | 2 | Data Extension Subheader definition, shall be "DE" |
| DESID | 25 | The identifier for this segment. For streaming, it shall always be |

| | | |
|---|---|---|
| | | equal to "STREAMING_FILE_HEADER" |
| Version of the Data Definition (DESVER) | 2 | Shall be a positive number, the minimum is 01 and maximum is 99. This version number is a registered value. |

Following these three fields are the security classification fields which are covered in this document as part of the security classification fields for segment subheaders. Following the DES Security Control Number (refer to classification fields), are new fields for the streaming subheader. The following fields are defined afterwards.

**Table 5-16. Data Extension Segment Subheader Fields for Streaming Header (ii)**

| Field | Size | Description |
|---|---|---|
| Length of DES-Defined Subheader Fields (DESSHL) | 4 | The Length of the next fields in this table. |
| Streaming File Header Length 1 (SFH_L1) | 7 | This value is equal to the number of bytes in the Streaming File Header Replacement Data field (SFH-DR) defined afterwards. |
| Streaming File Header Delimiter 1 (SFH_DELIM1) | 4 | This field is always 0x0A6E1D97, which serves as a unique starting delimiter prior to the SFHDR field. |
| Streaming File Header Replacement Data (SFHDR) | Equal to SFH_L1 and SFH_L2 (both have the same value) | Contains a new NITF file header up to the point until the last unknown size is implemented (all 9's). |

| | | |
|---|---|---|
| Streaming File Header Delimiter 2 (SFH_DELIM2) | 4 | This field is always 0x0ECA14BF, which serves as a terminating delimiter to the SFHDR field. |
| Streaming File Header Length 2 (SFH_L2) | 7 | Should be exactly the same value as SFH_L1. |

## PRODUCT: NITF 2.1

### LOCATIONS:

When this feature is used, the Streaming File DES (segment) is located at the end of the file, the final DES if there is more than one.

### CONCERNS AND RECOMMENDATIONS:

**Data Hiding -** Not having a DES subheader with the real header, or a delimiter, or having invalid lengths may introduce parsing problems and data may be hidden earlier in the undefined length segments.

**1** **Validate**: If all 9's are defined in the standard NITF File Header, check that this subheader exists at the end of the file by identifying "STREAMING_FILE_HEADER" in a "DE" subheader.

**2** **Validate:** Check that the delimiters can be found and that the SFHDR data is a correct NITF File Header.

**3** **Validate**: Check that both length (SFH_L1/SFH_L2) fields are correct and that SFHDR data is that size.

4 **Validate**: Perform the same validation checks as a standard NITF file header on the SFHDR field.

5 **Remove:** Remove the Streaming File DES Header and repack the data into a NITF file that is of known size.

**NITF.5.22: END**

## 5.5 Reserved Extension Segments

**NITF.5.23:RES SUBHEADER – INTRODUCTION**

**DESCRIPTION:**

The Reserved Extension Segment (RES) Subheader introduces three fields that define the beginning of this subheader. The remaining fields of this subheader such as classification information are defined for all subheaders earlier in Section 5. The following table defines the first three fields of this subheader.

**Table 5-17. Reserved Extension Segment Subheader Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| File Part Type (RE) | 2 | Shall be "RE" |
| Unique RES Type Identifier (RESID) | 25 | ISMC Registered values for an identifier. |
| Version of the Data Definition (RESVER) | 2 | Version number of the RES, which is also registered with the identifier |

**PRODUCT: NITF 2.1**

**LOCATIONS:**

The fields defined in this section follow the last DES block of data in the file.

**CONCERNS AND RECOMMENDATIONS:**

**Data Hiding -** Invalid values here may introduce hidden data.

1 **Validate**: Check that the File Part Type is RE.

2 **Validate**: Check that the Type Identifier and Version belong to a list of registered values by ISMC.

**NITF.5.23: END**

## NITF.5.24:RES SUBHEADER – USER DEFINED DATA

### DESCRIPTION:

At the very end of the RES Subheader there are three fields defined in the table below. It is entirely user-defined formatted data, but shall be registered by ISMC.

**Table 5-18. RES User Defined Data Fields**

| Field | Size (bytes) | Description |
|---|---|---|
| RES User-Defined Subheader Length (RESSHL) | 4 | Contains the number of bytes in the field RES User-Defined Subheader Fields (RESSHF) |
| RES User-Defined Subheader Fields (RESSHF) | Equal to value in field RESSHL | Contains user-defined fields remaining all alphanumeric characters. |
| RES User-Defined Data (RESDATA) | Variable, determined by previous field for user-defined data. | Binary or character data according to user specification. |

### PRODUCT: NITF 2.1

### LOCATIONS:

These three fields are located at the very end of the RES Subheader.

### CONCERNS AND RECOMMENDATIONS:

**Data Hiding and Data Attack -** User defined data in RES allows for binary or ASCII characters to be added.

**1** **Validate**: Check that the length of the User-Defined Data and Subheader fields are the appropriate length.

**2** **Remove**: Remove User-defined data and set the User-defined Subheader Length to 0. This may require realigning the file and changing lengths throughout.

**3** **Replace**: Replace User-defined data with all zeros and keep the User-defined Subheader length as is.

**Data Disclosure -** Information in user-defined data may contain sensitive text.

**4** **External Filtering Required**: Pass the RES User-Defined Data to an external filter; it may contain free-text.

## NITF.5.24: END

## 5.6 Tagged Record Extensions

### NITF.5.25:TAGGED RECORD EXTENSION (TRE) FORMAT

**DESCRIPTION:**

Tagged Record Extension (TRE) data is located throughout the NITF file in numerous extended data locations. They are located in the NITF File Header but can also overflow into a specific DES. There are two types of this TRE data: Controlled Extension (CE) and Registered Extension (RE). A CE is uniquely identified by the field CETAG, which is under complete control by the NTB. A RE also contains a unique identifier called RETAG, but its data and structure to this information is not managed by the NTB.

If a NITF file viewer does not implement either CE or RE data, it is instructed to ignore this information according to the standard. All of these fields are required fields in the NITF standard. The minimum length for REDATA or CEDATA is 1 byte, but it must exist. The segment itself can be deleted from where it is defined (described in Locations section below).

**Table 5-19. Tagged Record Extension Data**

| Field | Size (bytes) | Description |
|---|---|---|
| RETAG/CETAG | 6 | Alphanumeric identifier either of which is registered with ISMC. |
| REL/CEL | 5 | This field defines the length of the next data field. |
| REDATA/CEDATA | Equal to value defined in REL/CEL (previous field). | Variable length field with binary or ASCII user-defined data. |

An example of TRE data is shown below, which is included in the Extended Header Data section of the NITF File Header. The content of TRE appears in the file exactly as below:

```
JITCID00200I_3228C, Checks multi spectral image of 6 bands, the image subheader
tells the receiving system to display band 2 as red, band 4 as green, and band
6 as blue.
```

The value of each field for this example is shown in Table 5-20 below.

**Table 5-20. TRE Data Example**

| Field | Value |
|---|---|
| RETAG/CETAG | JITCID |
| REL/CEL | 00200 |
| REDATA/CEDATA | I_3228C, Checks multi spectral image of 6 bands, the image subheader tells the |

| | |
|---|---|
| (This is 200 bytes long as defined by REL/CEL, spaces follow the end of this field to reach 200 bytes) | receiving system to display band 2 as red, band 4 as green, and band 6 as blue. |

**PRODUCT: NITF 2.1**

**LOCATIONS:**
TRE data can exist in the NITF File Header User Defined Data (UDHD) field. TRE data can exist in the Image Subheader User Defined Image Data (UDID) field. TREs are also available in each Extended Subheader field (NITF File Header, Image Subheader, Graphic Subheader, and Text Subheader). A DES may contain overflowed TRE data in one of its segments if there is not enough room in the Extended Subheader. An overflow field in a subheader or NITF file header will identify the ID of the segment where the remainder of the TRE data exists.

**CONCERNS AND RECOMMENDATIONS:**
**Data Attack and Data Hiding -** Unknown identifiers might be ignored which can contain binary data (executable) or hidden sensitive text.

**1** **Validate**: Check that the RETAG/CETAG is on a valid list registered by ISMC.

**2** **Validate**: Check that the length of REDATA/CEDATA is accurate.

**3** **Replace:** Replace TRE data with a single byte space and change the length fields accordingly. This will require realigning the file and changing lengths throughout. This will negatively impact applications relying on TRE data.

**4** **Replace:** Replace all TRE data with spaces and keep length fields unmodified. This will negatively impact applications relying on TRE data.

**Data Disclosure -** User data may contain sensitive information.

**5** **External Filtering Required:** Pass the contents of REDATA/CEDATA to an external filter.

**NITF.5.25: END**

## 5.7 All Segment Data

**NITF.5.26: ALL SEGMENT DATA**

**DESCRIPTION:**
The data following each subheader is defined by its own format; this could be a specific image format, computer graphic, or formatted text such as United States Message Text Format (USMTF). Segment data is outside the scope of this document, with the exception of TRE data

which is covered in Section 5.6. All segment data can be extracted and filtered by an appropriate filter that is designed for that particular data format.

If the segment data is an image it may contain multiple blocks of images. In this case, there may be multiple files of image data located within the image segment.

### PRODUCT: NITF 2.1

#### LOCATIONS:

This data follows each subheader until the next subheader begins.

#### CONCERNS AND RECOMMENDATIONS:

**Data Attack, Hiding, and Disclosure -** Embedded content inherits many risks from that file format. Since the data could be of several types, it may introduce each type of risk.

**1** **Remove:** Remove segments (subheader and data) for unknown types of segment data.

**2** **External Filtering Required:** Pass entire segment data to an external filter.

**3** **External Filtering Required:** For blocked images, extract each block to an appropriate image filter, reconstruction of the entire image will require placing each same-size block back into place.

**4** **Review:** Review each segment individually and then as a whole combined with other segments.

**5** **Review:** For blocked images, review the entire set of blocks (entire NITF File).

### NITF.5.26: END

# 6. ACRONYMS

**Table 6-1 Acronyms**

| Acronym | Denotation |
|---|---|
| TRE | Tagged Record Extension |
| DES | Data Extension Segment |
| RES | Reserved Extension Segment |
| NITF | National Imagery Transmission Format |
| NSIF | NATO Secondary Imagery Format |
| BCS | Basic Character Set |
| USMTF | United States Message Text Format |
| ECS | Extended Character Set |
| JPEG | Joint Photographic Experts Group |
| ASCII | American Standard Code for Information Interchange |
| MRI | Magnetic Resonance Imagery |
| CAT | Computerized Axial Tomography Scan |
| MONO | Monochrome |
| RGB | Red, Green, Blue |
| LUT | Look-Up Table |
| FIPS | Federal Information Processing Standard |
| PUB | Publications |
| ISMC | Imagery Standards Management Committee |
| CGM | Computer Graphics Metafile |
| GENC | Geopolitical Entities, Names, and Codes |

# 7. REFERENCED DOCUMENTS

[1] MILSTD-2500C. NITF 2.1. Online reference available at: http://www.gwg.nga.mil/ntb/baseline/docs/2500c/index.html

[2] NITFS History, in the Bandwidth Compression Symposium, Arnold, MO. 15 May 2002. Available at: http://www.gwg.nga.mil/ntb/2002SICS/02NITFhist.PDF