ADatP-3 Part I

NATO MESSAGE TEXT FORMATTING SYSTEM (FORMETS)

SYSTEM CONCEPT AND DESCRIPTION

May 1994[1]

[1] Including Editorial Changes agreed by the MTFWG

This page is intentionally blank

RECORD OF CHANGES

Change Date	Date Entered	Effective Date	By whom entered
15 Sep 1997	17 Jun 1999	15 Sep 1997	VC MTFWG

This page is intentionally blank

TABLE OF CONTENTS

CHAPTER 6

MESSAGE TEXT LEVEL CONVENTIONS

LIST OF FIGURES

CHAPTER 1

GENERAL INFORMATION

101. Purpose of the System

Standardisation of MESSAGES used for information exchange will improve interoperability between different national and NATO authorities and systems. To that end, the NATO Message Text Formatting System (FORMETS) provides the rules, constructions and vocabulary for standardised CHARACTER-oriented MESSAGE TEXT FORMATS (MTF) that can be used in both manual and computer-assisted operational environments. FORMETS is specified in Allied Data Publication Number 3 (ADatP-3).

102. Policy of Use

a. FORMETS is to be used for all formatted character-oriented messages within the NATO Command, Control and Information System unless specifically excluded by multinational agreement. It is concerned solely with that part of a message that contains the information the originator wishes to communicate. The transmission of FORMATTED MESSAGES remains in accordance with the instructions given in relevant Allied Communications Publications (ACP).

b. FORMETS may also be used for other NATO and national information systems.

c. ADatP-3, Part I, is intended primarily for use by personnel involved in the development and implementation of message text formats. It may also contribute to the training of personnel involved in using those standards.

d. Recommendations for expanding and enhancing FORMETS should be generated by national and Major NATO Commander (MNC) usage.

103. Subdivision of ADatP-3

ADatP-3 is subdivided into five parts as follows:

a. Part I provides a detailed description of FORMETS. This part is subject to ratification by the nations.

b. Parts II to IV provide catalogues of message text formats, SET FORMATS and FIELD FORMATS respectively. Indexes and cross reference listings of formats are included in each part to facilitate the use and management of approved standards. These parts of the publication are dynamic, liable to regular and significant additions, amendments and deletions. It is sufficient that changes to Parts II to IV are multi-nationally agreed by the appropriate NATO body prior to promulgation for general implementation and operational use.

c. Part V provides a Keyword-Out-of-Context (KWOC) Directory intended to help developers determine whether an approved standard pertaining to a specific subject is available. The directory is in two sections, a KWOC listing of field format subjects and a KWOC listing of set format subjects. The same management principles that apply to Parts II to IV are applied to Part V.

104. Annexes and Glossary to Part I

There are four annexes that formalize or complement descriptions provided in the main body of Part I and a glossary. The glossary of terms and definitions is particular to this publication. Although the usual practice of explaining the meaning of a term at the point in the text where it first occurs is normally followed, this is not feasible in all cases. Consequently, occasional reference to the glossary may be necessary. When a term in the glossary first occurs in the text, it is printed in capital letters. If any discrepancies are discovered between different annexes, between an annex and the main body, or within the main body itself, a nation or MNC will generate a change proposal to create consistency. The discovery of an inconsistency will result in agreed interim "Case Law" guidance which will be incorporated in the next change to Part I. The four annexes are as follows:

a. Annex A. SYNTAX OF THE NATO MESSAGE TEXT FORMATTING SYSTEM.

This annex provides a compilation of syntactical statements which is intended to ensure the unambiguous application of the rules governing the design and the use of message texts. The annex has equal precedence with the main body of Part I.

b. Annex B. FIELD FORMAT SPECIFICATIONS.

This annex provides instructions, in addition to those found in the main text of Part I, as to how field formats shall be documented in Part IV.

c. Annex C. STRUCTURAL RELATIONSHIPS.

This annex provides the specifications of computer processable notations by which the definition of the structure of each message can be completed by the consistent application of the rules governing the form and content of messages as constructed of SEGMENTS, SETS and FIELDS.

d. Annex D. ABBREVIATIONS.

This annex provides abbreviations used in this publication.

105. Use of Examples

Examples are used throughout Part I to demonstrate FORMETS principles. These examples are not necessarily approved message text formatting standards and should not be considered as such during the ratification by nations or subsequent use of the document. Changes to the examples will be required only when they no longer properly demonstrate the associated FORMETS principles.

106. Responsibility

NATO C3 BOARD, INFORMATION SYSTEMS SUBCOMMITTEE, MESSAGE TEXT FORMAT WORKING GROUP is responsible for the development and maintenance of FORMETS.

107. Submission of Change Proposals

Proposed changes to ADatP-3, Parts I through V should be submitted in accordance with the ADatP-3 Configuration Management Plan to:

Vice Chairman MTFWG
NATO HQ C3 STAFF
NATO Headquarters
B 1110 Brussels
Belgium

CHAPTER 2

SYSTEM CONCEPT

201. The Requirement

The exchange of information is an essential activity in any defence organisation. The information to be transferred should be concise, accurate, up to date and readily understandable. A standardised information exchange language must possess these characteristics.

202. The Characteristics of a Suitable Language

a. For messages, a natural language in a normal free text mode can be used. However, this procedure, though exploiting the flexibility of a natural language, has serious disadvantages in that the free text messages often prevent rapid processing. Sentences often contain words that can be omitted without loss of information, and words frequently have more than one meaning. These disadvantages become even more obvious when computer assistance to message processing is considered.

b. To achieve a better solution for messages, two separate actions can be taken. First, create an artificial language with a vocabulary restricted to words (including abbreviations and codes) for which unambiguous meanings have been agreed by all participants. This is especially desirable in a community that uses multiple natural languages. Second, create structures for this artificial language so that as much information as possible is conveyed purely by the position of words within predetermined formats to complement the information provided by the words themselves.

c. The requirement for an artificial language for messages is not dependent upon the existence of computer assistance to message processing. However, a computer can be more efficiently programmed to process information presented in an artificial language, that uses only precisely defined words and predetermined formats, than to process information in a natural language. This is an important and significant additional benefit.

203. The Role of FORMETS

FORMETS defines an artificial language suitable for the exchange of NATO defence information. The system comprises the rules governing representation of agreed conceptual definitions and arrangement of these representations within predetermined formats. The application of FORMETS rules to information exchange requirements results in open-ended inventories of agreed representations, i.e. field formats, set formats and message text formats. FORMETS, as guided by the NATO Interoperability Management Plan, contributes to and draws from other interoperability standards, such as agreed NATO terminology, to the optimum extent achievable.

CHAPTER 3

FORMETS GENERAL DESCRIPTION

301. Purpose

The purpose of this chapter is to provide a general description of FORMETS by establishing:

a. The relationships between formatted message text components within a typical message.

b. The correlation between the architectures of formatted and natural language message texts.

c. The semantic relationships of the artificial FORMETS language with a natural language.

d. The basic syntax of FORMETS components.

e. The basic rules that affect FORMETS components.

f. The precedence of FORMETS components documentation.

302. Overall Message Composition

Almost all forms of formatted messages have a heading, a text and an ending. The heading and ending portions of a message are dependent upon protocols for the communications system being used. Procedural guidance for these portions is specified in the appropriate communications publications and is not affected by message text formatting rules as laid down in this document.

303. Text Composition

In FORMETS the text, as shown in Figure 3 - 1, is divided into three portions to account for additional communications, security or other requirements that must be met. These divisions are:

```
                           ROUTINE
                           R 061200Z MAR 93
                           FM SHAPE
                           TO AIG 6027
                           BT
Introductory Text          NATO UNCLAS
                           SIC XYZ
                           EXER/CMX 93//
                           MSGID/LOGHOLDLAND/SHAPE/001/MAR//
                           REF/A/SHAPE/051200Z MAR 93//
                           EFDT/121500Z/MAR//
                           PART/1//
                           GENTEXT/LOGISTIC HOLDINGS/NONE//
                           COVAL/-/-/6/GE/-/ARMR/DIV/A/14/ATTACK//
                           GENTEXT/PROBLEM AREAS/NONE//
Main Message Text          PART/2//
                           ORGID/-/144/GE/-/ARMR/BM/A//
                           1FWPNS
                           /CODE  /NAME        /QTY-OH/QTYSVC/QTYTOE
                           /ABC123/MARDER      /    37/    32/    50
                           /XYZ789/120MM MORTAR/     3/     3/     6//
                           5POL
                           /CODE  /POL-TYPE    /CBM-OH   /DOS
                           /DSL456/DIESEL      /      800/   2
                           /HYDFLU/HYDRAULICS  /       20/   2//
                           RMKS/THIS FREE TEXT SET IS ADDED FOR DEMONSTRATION PURPOSES ONLY//
                           CLOSTEXT
Closing Text*              .
                           BT
```

*   If required, closing text information begins on the line immediately after main message text.

Figure 3 - 1   Example of a Formatted

a. Introductory Text

The INTRODUCTORY TEXT is reserved for information required by procedures established for the communications system in use. It often contains, but is not limited to, the overall security classification of the message, flag-words, special handling instructions, message distribution information, and, when a lengthy message has been transmitted in sections, message section identification. The structure of introductory text information is not governed by message text formatting rules and, therefore, is not included in the message text formats provided in Part II.

b. Main Message Text

The MAIN MESSAGE TEXT contains all the information concerning the subject being addressed by the formatted message. It is structured in accordance with a message text format provided in Part II and governed by message text formatting rules. The main message text may be interrupted, where called for by applicable communications procedures, to enable insertion and transmission of information concerning page numbering, message sectioning or similar communications-system-related requirements. As with the introductory text, this inserted information is not governed by message text formatting rules.

For brevity in the remainder of this document main message text will be referred to as MESSAGE TEXT.

c. Closing Text

The CLOSING TEXT provides information that is related to, but subjectively not part of, the message text. It also provides for unique national requirements that cannot be accounted for within message text.

304. Structural Components

The establishment of structural components is fundamental to any language. The message text of the FORMETS artificial language is constructed from fields and sets in much the same way that information is logically conveyed by a text constructed from words and sentences of a natural language. In this artificial language, a field, set and message text are similar to a word, sentence and text, respectively, in a natural language.

305. Field Structure

In FORMETS each field consists of a FIELD MARKER (/) followed by the FIELD CONTENT. The field content comprises either a DATA CODE with a FIELD DESCRIPTOR if specified, or a single hyphen character. BLANK CHARACTERS may be included as explained in paragraphs 313.d, 410.b,and 503.g.

306. Set Structure

A set consists of an identifier, COLUMN HEADER(S) in the case of a columnar set, field(s), and an END-OF-SET MARKER (//) as explained in paragraphs 312.b, 502 and 503. The identifier is a group of characters called the SET FORMAT IDENTIFIER, which serves as a key to the structure and information content of each set.

307. Message Text Structure

A message text is a series of sets. A MESSAGE TEXT FORMAT IDENTIFIER, is specified for each message text format and serves as a key to the structure and information content of the message text format. This identifier is entered in the first field of the Message Identifier set that is included in every formatted message text. Figure 3 - 2 provides a summary of structural terms and definitions employed within FORMETS.

308. FORMETS Vocabulary

The FORMETS vocabulary comprises message text format identifiers, set format identifiers and data codes. Specifications for these groups of words are included in:

a. Part II of this publication for message text format identifiers.

b. Part III of this publication for set format identifiers.

c. Part IV of this publication for data codes.

Field	**Set**	**Message Text**
A field is a unit of information in a set. It is made up of a field marker followed by the field content.	A set is a grouping of related fields. It is made up of a set identifier, column header(s) in columnar sets, and field(s) followed by an end-of-set marker.	A message text is a group of related sets that are transmitted together as a message.
Field Format	**Set Format**	**Message Text Format**
A field format is a specification of one or more fields that are to contain similar data. It consists of a reference number, name, definition, field use designators, data items and codes, and associated structures.	A set format is a specification of a set. It consists of a set name, set identifier, and the identifiers of the field formats and field use designators of those fields to be included in the set.	A message text format is a specification of a message text. It consists of a message name and identifier and the identifiers of the set formats of those sets to be included in the message text in a prescribed order.

Figure 3 - 2 Summary of Structural Terms and Definitions

309. Use of Natural Language

There is also a requirement to exchange information that can only be expressed in natural language. For this purpose, FORMETS provides the FREE TEXT FIELD in the FREE TEXT SETS.

310. Semantics of FORMETS

Like natural language, FORMETS requires semantic principles in order for terms to provide context meaning. In formatted messages, this context meaning is pre-established. A message text format defines the context in which its sets are used, and the set format establishes the context in which its fields are used. Message text format identifiers and set format identifiers are the words used to convey the context meaning at the message and set levels, respectively. Context meaning of fields is established by specifying predefined field uses to the FIELD POSITION of sets. Each field format has one or more field uses. For example, **Date Time of Sighting** and **Date Time of Incident** may be two FIELD USE DESIGNATORS (FUD) of a field format named **Date Time**. The same data code might be appropriate for use in both cases, but the context meaning differs. In formatted messages, therefore, the meaning of a data code is established by its position in a set, and its full context meaning is derived from the use of the set in a message text.

311. Syntax of FORMETS

Syntactical rules govern the structure and the arrangement of the components of FORMETS. Annex A contains a formal description of the FORMETS syntax using Backus Naur Form (BNF). It should be understood, however, that the message text is influenced by procedures established for the communications system being used. Instructions contained in ACP 121, ACP 127 and their supplements, for example, require specific information, such as security classification, to be included in the message text. In addition, the message text may be interrupted, when called for by applicable communications procedures. This type of information is neither governed nor precluded by FORMETS.

312. Basic Structure of Formatted Messages

The message text of a formatted message is composed in accordance with the message text format that prescribes an ordered arrangement of sets. A set format, in turn, prescribes an ordered arrangement of fields. A field format prescribes the values from which a message drafter may choose in populating that particular field. These three types of formats (message text, set, and field) provide the basic hierarchical relationships that exist among all components of a formatted message.

a. Message Text Formats

A message text format specifies a sequence of set formats and has a unique identifier. This identifier is preferably a mnemonic keyword that associates with the type of information that is to be provided by the message text. It also cues the human and the computer to the sets allowed by the message text format and the order in which they must be arranged. The set formats with identifiers EXER, OPER, MSGID, and REF occupy, in this sequence, the first SET POSITIONS in each MTF to the extent that they are specified. Of these set format identifiers only MSGID must be specified in each MTF. Note that the EXER and OPER sets are mutually exclusive. Use of one or the other is only permitted when operationally appropriate, that is, if the message relates to a specific exercise or operation, otherwise neither is used. A set position is defined to be the sequence number, starting with 1, in which a set format is specified as a component of a message text format.

The remaining set positions of the message text are allocated to set formats as needed to satisfy a particular information requirement. Message text formats are assigned a MESSAGE TEXT FORMAT NAME and are documented in Part II where they are identified by a unique MESSAGE TEXT FORMAT INDEX REFERENCE NUMBER and by a unique message text format identifier. Message text level conventions are described in detail in Chapter 6.

b. Set Formats

A set format specifies a sequence of one or more field formats and, if used, their respective field descriptors and has a unique set format identifier and a unique SET FORMAT NAME. The set format identifier is a keyword (preferably mnemonic) that distinguishes the set format from all others. A field position is defined to be the sequence number, starting with 1, in which a field format is specified as a component of a set format. The set formats to be used in a particular message text format are identified in Part II and specified in Part III. Set level conventions are described in detail in Chapter 5. There are three types of sets:

(1) Linear Set

A LINEAR SET is a set having one or more fields presented in a horizontal arrangement. The following is an example of a linear set:

```
MSGID/GREEN/CTG89.05/895001/MAR/REQ/1//
```

(2) Columnar Set

A COLUMNAR SET is a set having a group of fields that provide an ordered arrangement of data aligned vertically under a horizontal array of column headers. The following is an example of a columnar set:

```
2AMMOH
/CODE  /AMMO-NAME   /QTY-OH   /DOS
/AMMO50/50CAL       /     3000/   2
/HOW105/105MM       /      600/   2//
```

(3) Free Text Set

A free text set is a set that provides the capability to include explanatory, unformatted information in a formatted message. The following is an example of a free text set:

```
NARR/THE ABOVE SETS ARE FOR DEMONSTRATION PURPOSES ONLY//
```

c. Field Formats

A field format (FF) specifies the allowed data codes for all fields using a class of subject related data. A field format may support either elemental concepts, such as the name of a location, or a combination of elemental concepts, such as date-time group, that are linked by common usage or formatting requirements. The field format, therefore, is the fundamental building block in the overall structure of formatted messages.

A field format is assigned a unique FIELD FORMAT NAME and a unique FIELD FORMAT INDEX REFERENCE NUMBER (FFIRN). Since all of the DATA ITEMS and data codes associated with a given field format may not be appropriate for use in all fields that use that format, one or more FUDs are specified for each field format. Only the data items and codes appropriate for the given use are assigned to a FUD. Each FUD is identified by a FIELD USE DESIGNATOR NUMBER (FUDN) which is unique within the field format. In Part III each field position of every set format is assigned the FFIRN/FUDN that identifies the FUD and thus the legal set of data items and codes for the field. Field level conventions are described in detail in Chapter 4.

313. Basic Structural Rules Affecting Use of FORMETS Components

a. All fields in linear and columnar sets are formatted; that is, they have a specified structure. Free text sets contain a free text field whose length is not specified.

b. All fields begin with a field marker, which is a single slant character (/). This is the only allowed use of a single slant character in formatted (linear or columnar) sets. However, the single slant character may be used as part of the text in the free text field of free text sets in addition to being the field marker used in these sets.

c. Field contents of linear sets consist of a data code, or a field descriptor and a data code, or a single hyphen character.

d. Field contents of a columnar set are as for those of linear sets except that blank characters may be added as required for positioning and justification.

e. All sets begin with a set format identifier, have one or more fields and terminate with an end-of-set marker, which is two consecutive slant characters (//). This is the only permitted use of consecutive slant characters. Note that this precludes the use of single slant characters as the first or last character of the free text field of a free text set since such use would create consecutive slant characters.

f. All sets start at the left margin; that is, the first character position of the message LINE.

g. A line may contain up to 69 characters (excluding the code sequence causing printing to continue at the start of the next line - the new line code), including those used for the set format identifier, field markers, field contents, and the end-of-set marker.

h. Sets may be continued onto a second line or subsequent lines, but with the following stipulations:

(1) The set format identifier is not repeated on the continuation line(s).

(2) The end-of-set marker is placed only after the final completed field of the set.

(3) Formatted fields are not divided between lines. Continuation lines for linear sets begin at the left margin with the field marker for the next field in the set. The following example shows how a linear set is continued:

```
REF/A/EN BATTLE EQUIP REP/USS JOHN F KENNEDY/061200Z9MAR86/1234567
/PASEP/XYZ//
```

(4) The formatted fields of columnar sets are repeatable as a group. Each repetition of the group starts on a new line, occupies only one line, and may not be continued on the next line. The following example shows how columnar sets are continued for each occurrence of the repeatable group of fields:

```
5POL
/CODE  /POL-TYPE    /CBM-OH   /DOS
/DSL456/DIESEL      /      800/   2
/HYDFLU/HYDRAULICS  /       20/   2//
```

(5) The free text field in free text sets may be divided between lines. Continuation lines for free text fields within free text sets begin at the left margin. Occasionally, blank characters or empty lines may be used to space free text for clarity of presentation. A slant character within the free text does not constitute a field marker. The following example demonstrates how the free text field in a free text NARRATIVE SET is continued:

```
NARR/THIS IS AN EXAMPLE OF A FREE TEXT SET THAT SHOWS THAT THE FIRST
LINE MAY NOT CONTAIN MORE THAN 64 CHARACTERS AFTER THE FIELD MARKER,
BUT SUBSEQUENT LINES (WHICH ARE NOT STARTED BY AN IDENTIFIER OR FIELD
MARKER) MAY CONTAIN UP TO 69 CHARACTERS//
```

i. The end-of-set marker may not be split between lines. Thus, a continuation line may occasionally contain only the end-of-set marker. In the following example, all 69 character positions of the first line have been used for the set format identifier and fields of the set:

```
AREA/3014N8-08141W4/3200N5-07600W3/3020N5-07200W9/NOVEMBER XRAY TANGO
//
```

j. Appropriate free text sets may be inserted between sets, but they must not interrupt a set. For instance, in a columnar set with four lines, explanatory information regarding data in the third line would be entered in an free text AMPLIFICATION SET positioned immediately after the completed columnar set, as demonstrated in the following example:

```
5POL
/CODE  /POL-TYPE    /CBM-OH   /DOS
/DSL456/DIESEL      /      800/   2
/HYDFLU/HYDRAULICS  /       20/   2//
AMPN/ADDITIONAL INFORMATION REGARDING ANY LINE ENTRY ABOVE COULD BE
REPORTED IN THIS FREE TEXT SET BEGINNING ON THE LINE IMMEDIATELY
FOLLOWING THE 5POL SET//
```

314. Repetition

Set and field formats may be designated as being repeatable within their containing formats, subject to the rules and conventions prescribed in Chapters 5 and 6.

315. Segmentation

Two or more sequential set positions in an MTF that are related by their content can be designated as a segment. Such a segment may, unless restricted by the MTF, occur as many times as are needed within a message. Segments are therefore said to be repeatable. Two or more sequential set positions that are completely within a designated segment may also be identified as a segment. In this manner segments can be nested in the MTF to whatever extent is required.

316. Occurrence Categories

Every segment in a message text format is assigned an OCCURRENCE CATEGORY based on the importance and logical relevance of the related information to the purpose being served by the message. Every set position in a given segment is assigned an occurrence category based on the importance and logical relevance of the set to the purpose being served by the segment. For those set positions that are not contained in a segment, the occurrence category refers to the importance and logical relevance of the set within the message itself. Every field position in a set format is assigned an occurrence category based on the importance and logical relevance of the field to the information to be conveyed by the set. The occurrence category of the field positions within a set format will remain constant regardless of the message text format in which the set is used. The occurrence categories, explained below, are assigned by the format designer. Specific rules concerning their application and use are given in Chapters 5 and 6. In the following descriptions of occurrence categories the term "format position(s)" includes segment(s) as well as set and field format position(s).

a. Mandatory

Format positions that provide information considered essential to the purpose of their containing formats are assigned a MANDATORY occurrence category. For example, a field position that is essential to a set format, a set position that is essential to a segment, and a segment that is essential to a message text format are all assigned a mandatory occurrence category. This category is designated by the character M in the containing set format or message text format.

b. Operationally Determined

Format positions whose use is determined solely by operational considerations are assigned an occurrence category of OPERATIONALLY DETERMINED. The use of format positions with this occurrence category shall have no structural dependence upon any other part of the message text. That is, the message text is structurally legal with or without the format position without regard to any other information in the message text. This category is designated by the character O in the containing set format or message text format.

c. Conditional

Format positions whose use is dependent upon some other discernible feature of the message text are assigned an occurrence category of CONDITIONAL. This category is designated by the character C in the containing set format or message text format. A format position with this occurrence category has its use governed by a condition requiring or prohibiting the use of the format position. The condition that governs the use of the format position must be documented in that containing format which allows the internal verification. If the condition is not met, that is, in those cases where its effect is neither to require nor to prohibit the format position, the treatment of the format position is as though it had an occurrence category of operationally determined (O).

The occurrence category C can only be used to indicate MTF internal structural dependencies; availability of information or the presence of external situations or events are not to be used as conditions governing the use of a format position under this occurrence category. (See Annex C for the specification of the conditions and their effects.)

The only discernible features of the message that can be used for the purpose of a condition are:

(1) the use or omission of a non-mandatory segment, set position, or field position.

(2) the number of occurrences of a repeatable segment, set, field group, or field.

(3) the content of a field subject to the following rules:

(a) The field descriptor is not to be included as part of the field content in the condition, but, when present within the field, serves to identify which ALTERNATIVE FIELD FORMAT (AFF) is used.

(b) The field content to be used in the condition may be the data code or the FUD selected for use in the field.

(c) Only reference to whole fields is permitted, that is, reference to one or more of the COMPONENT FUDs that make up a COMPOSITE FUD is not permitted.

317. Precedence of FORMETS Components Documentation

Due to the complexity and interrelated nature of FORMETS components, it is inevitable that some documentation variances will exist. The precedence of FORMETS documentation set forth below will be applied to resolve such variances

a. Part I shall have precedence over all other parts.

b. Part IV shall have precedence over any other specification of field format attributes. Within Part IV, the precedence of field format attributes is as follows:

(1) Specification of data items and data codes.

(2) Specification of field uses of an ELEMENTAL FIELD FORMAT.

(3) Specification of an elemental field format.

(4) Specification of field uses of a COMPOSITE FIELD FORMAT.

(5) Specification of a composite field format.

c. Part III shall have precedence over any other specification of set format attributes.

d. Part II shall have precedence over any other specification of message text format attributes.

CHAPTER 4

FIELD LEVEL CONVENTIONS

401. Data Item and Data Code Specification

Words of natural language are not always suited for use in formatted messages. These words can usually be represented either by abbreviations without loss of mnemonic value, or by codes already in existence that are accepted through general use. Specification of FORMETS data codes is provided in Part IV as part of the field formats with which they are associated. The field formats contain listings of relevant data items and the data codes used to represent those data items. Data codes can contain any ALPHANUMERIC CHARACTER. If a blank character is used in a data code it must be embedded. Thus a data code may not begin or end with a blank character. However, a single blank character or a single hyphen (-) character is permitted for use as a separator in a composite field format. (See paragraph 402.c for a description of composite field formats.) A data code must not contain more than 68 characters. While it is possible to list every data item and associated data code permitted for each field format, in order to effect economies for the convenience of both the human and computer use, the conventions of data item and data code specification in Part IV must be adhered to in accordance with the rules given in the following subparagraphs and as further detailed in Annex B.

Data items and data codes are specified in Part IV by means of the three types of specification entries prescribed below:

a. Individual Entry Type.

This entry type specifies an individual data item and data code.

b. Range Entry Type.

This entry type specifies a range of data items and data codes. There are three kinds of range entry:

(1) Integer Ranges.

(2) Ranges Containing Decimal Point Values.

(3) Alphabetic and Alphanumeric Ranges.

c. Instructive Entry Type.

This entry type specifies any data item and data code entry other than individual or range entries. Such an entry is instructive or informative in some way.

If an instructive entry type specifies a literal data item, this indicates that the FUD(s) using the entry can employ any data items that logically can be inferred from the field format definition, as modified by the explanation for the applicable FUD, and any entry in the Data Items and Codes Explanation column applicable to the FUD and the FF Remarks section, if present. The associated data code must also be literal. This indicates that any data code whose structure corresponds to that specified for the applicable FUD may be used.

The use of literal data items and data codes is to be limited to three circumstances described below.

(1) The literal data item and data code is to be used for fields that are to contain text, the exact wording of which cannot be anticipated.

(2) The literal data item and data code is to be used in those exceptional cases when the volatility of individual entries or ranges is such that the standard cannot be updated in time to reflect the operational needs.

(3) The literal data item and data code is to be used if the use of individual entries or ranges would result in a classification of Part IV that is higher than acceptable at the time of specification.

402. Field Formats

A field format is the specification, in Part IV, of the informational content of one or more fields that are to contain similar data. A particular use of a FF requires the specification of a field use designator. There must be at least one FUD specified for each FF. Each FUD is assigned one or more of the data items and associated data codes that have been specified for the FF. In this manner, the assignment of a FUD to a particular field in a set establishes the legal values that may be used in the field. Because of the need to concatenate data codes together as described in paragraph 407 it is necessary to define different types of FFs and FUDs. These types and their relationships are described below.

a. Elemental Field Format

An elemental FF is one in which data items and their associated data codes are specified.

b. Elemental Field Use Designator

An ELEMENTAL FIELD USE DESIGNATOR is a particular use of an elemental FF. It uses either all or some subset of the data items and associated data codes of the parent elemental FF.

c. Composite Field Format

A composite FF is one for which no data items and data codes are specified. Instead, a sequence of elemental FUDs is specified from which data items for the composite FF can be inferred. The data items associated by inference with a composite FF consist of all possible data items that can be constructed by combining each permitted data item from one elemental FUD with each permitted data item from the other elemental FUDs that make up the composite FF.

d. Composite Field Use Designator

Like the elemental FUD, the composite FUD is a particular use of the parent FF. Unlike the elemental FUD, subsetting of the data items and data codes for a particular FUD is not allowed. That is, each composite FUD permits use of all possible data items and data codes resulting from the concatenation of the data items and data codes of the elemental FUDs that make up the composite FF.

e. Component FUD

A component FUD is an elemental FUD used in the make-up of a composite FF. Note that a composite FUD may not be used as a component FUD. That is, each component FUD specified in the make up of a composite FF must be an elemental FUD.

403. Field Format Specification

A field format has been defined as a specification contained in Part IV. That specification includes the following elements which are further defined in Annex B:

- Field Format Index Reference Number.
- Field Format Name.
- Field Format Structure.
- Field Format Definition.
- Field Format Data Items and Data Codes.
- Component Field Use Designators.
- Field Format Remarks.
- Field Use Designator Specification.

404. Field Use Designator Specification

The field use designator specification is a part of the field format specification in Part IV. That specification includes the following elements which are further defined in Annex B:

- Field Use Designator Number.
- Field Use Designator Name.
- Field Use Designator Data Items and Data Codes.
- Field Use Designator Structure.
- Field Use Designator Explanation.

405. Field Format Structure and Field Use Designator Structure

A FIELD FORMAT STRUCTURE indicates the structure of all of the data codes associated with a field format by designating the number and permitted types of characters. However, since the contents of a field are defined by a FUD, which may represent a subset of the field format data codes, there is a FIELD USE DESIGNATOR STRUCTURE associated with each FUD. For example, a field format structure may be specified as "3-5AN" indicating that all of the codes associated with the field format can be expressed in three to five ALPHABETIC CHARACTERS and/or NUMERIC CHARACTERS. However, a particular FUD may use only the codes that are alphabetic. The FUD structure of that FUD would then be "3-5A". Character type symbols are defined as follows:

A =	Alphabetic Character [A,B,...Z]
B =	Blank (or space) Character
N =	Numeric Character [0,1,...9]
S =	SPECIAL CHARACTER [.,-()?]
X =	Alphanumeric Character (Alphabetic, Numeric, Special or Blank Character)

The structures for composite field formats consist of the FUD structures of the component FUDs that make up the composite field format. In the example of geographic coordinates (paragraph 407), the composite field format structure is 11AN. This comprises the FUD structures **2N**, **2N**, **1A**, **3N**, **2N**, and **1A** of the component FUDs. Since all FUDs of a composite field format utilize the same component FUDs the composite field format structure is identical to the combined FUD structures of each of the FUDs of the composite field format.

406. Field Format Types

Fields formats are fixed-length, variable-length, or free text depending on the requirements of field content. (Character position numbers in the following examples are provided only to indicate data code length.)

a. Fixed-Length Field Format

A field format that requires the same fixed number of characters for all data codes is called a FIXED-LENGTH FIELD FORMAT. In the following examples, the lengths of permitted data codes are exactly three and seven character positions, respectively.

```
(1)    FFIRN/FUDN                              1004/001     (3A)
       Character Position Number:       123
       Field:                         /APR

(2)    FFIRN/FUDN                              2000/002     (7AN)
       Character Position Number:       1234567
       Field:                         /051450Z
```

b. Variable-Length Field Format

A field format that specifies that the number of characters used to express data codes may vary between prescribed limits is called a VARIABLE-LENGTH FIELD FORMAT. The following examples demonstrate the entry of data codes associated with variable-length field formats, as used in linear sets:

```
(1)    FFIRN/FUDN                              1036/002     (1-6 A)
       Character Position Number:       123456
       Field:                         /SSBN

(2)    FFIRN/FUDN                              1008/003     (2-3A)
       Character Position Number:       123
       Field:                         /DAY

(3)    FFIRN/FUDN                              2061/001     (2-7ANS)
         composed of        1023/019 (1-5NS) and 1008/004 (1-2A)
       Character Position Number:       1234567
       Field:                         /105FT

(4)    FFIRN/FUDN                              1083/002     (1-11NS)
       Character Position Number:          12345678901
       Field:                         /PRF:245.5
```

In these examples, the length of each field could fluctuate between a minimum and maximum number of characters. In example (1), the FUD structure (1-6A) can accommodate one, two, three, etc., up to a maximum of six alphabetic characters. The character position numbers above each field demonstrate the difference between the length of the chosen field content and its maximum permissible length. Note that, in all but the second example, the field uses less than the maximum number of allowed characters; blank characters or leading zeros are not added to the field to make the completed field correspond to the maximum length. In contrast to the other examples, the character positions in example (4) are numbered beginning immediately after the colon. In this case, **PRF:** is a field descriptor added to the field content. Field descriptors are discussed further in paragraph 409 and in paragraph 506.

c. Free Text Field Format

A field format with no prescribed length limitation and allowing all permitted character types, except successive slant characters, is known as a free text field format.

407. Data Codes

A data code represents either a data item or a chain of linked data items. Normally, a data code may not begin or end with a blank character. The only exception is a single character code associated with a component FUD. (However, blank characters may precede or follow a data code for purposes of tabular presentation.) Forty-five characters are available for use in formatted messages. They are, by type, 26 alphabetic characters [A through Z], 10 numeric characters [0 through 9], 6 special characters [.,-()?], the slant character, the colon character and the blank (space) character. Two characters , slant [/] and colon [:] have specific formatting significance. The slant character is used as field marker. Two consecutive slant characters are used as end-of-set marker. The colon is used as part of the field descriptor, which is further explained in paragraph 409 and in paragraph 506. The colon character and non-consecutive slant characters may be used in the free text field of free text sets. No other use of the slant or the colon character is permitted. The hyphen [-], when used alone, indicates that the information to complete a field is either not available or not appropriate, as explained in paragraph 410 and in paragraph 411. The following are examples of data codes:

APR is a data code that represents the data item **APRIL**

JONES is a literal data code that represents an individual's name.

5211N14428W is a data code representing the data items that comprise geographic coordinates.

In these examples **APR** and **JONES** are data codes for ELEMENTAL FIELDS in that each represents an individual data item. **5211N14428W** is a data code for a COMPOSITE FIELD. It is made up of the six data codes **52**, **11**, **N**, **144**, **28** and **W**.

408. FUD Structure Application

The role of FUD structures is consistent throughout their application in different types of sets. In each case, the FUD structure accounts for the number and type of characters that comprise permitted data codes. However, the way in which data codes are presented in fields varies between set types, as discussed below.

a. Linear Set Application

Fields in linear sets are presented in a contiguous, linear arrangement. For example:

```
EFDT/071200Z/MAR//
```

Since there is no requirement for data separation within the fields, the field contents for linear set applications are related directly, and only, to the characteristics of the data codes, whether elemental (**MAR**) or composite (**071200Z**).

b. Columnar Set Application

Fields in columnar sets are arranged so that data codes appear in tabular form under appropriate column headers. For example:

```
2AMMOH
/CODE  /AMMO-NAME   /QTY-OH   /DOS
/AMMO50/50CAL       /    30000/   2
/HOW105/105MM       /      600/   2//
```

Note that most field contents have blank characters added to the data codes to provide the desired tabular alignment. In some cases, the blank characters follow the data codes (50CAL   ); in others, the blank characters precede the data codes (30000). These data codes are, respectively, LEFT-JUSTIFIED and RIGHT-JUSTIFIED. The number of character positions assigned to each field position of a columnar set and the justification of data codes within the fields are prescribed by the set format and explained further in Chapter 5.

c. Free Text Set Application

Free text sets contain one field of free text field format. This free text field is presented in a contiguous, linear arrangement and may be continued on one or more subsequent lines. Continuation lines for free text fields begin at the left margin. Blank characters or empty lines may be used to space free text for clarity of presentation. The free text field format is the only field format type that allows continuation of a field on one or more subsequent line(s). For example:

```
AMPN/USS GEORGE WASHINGTON WILL BE OPERATING IN THIS AREA DURING
NIGHT OF 15/16 JUNE//
```

Note that in this example there is but one free text field, that it extends to a second line, and that the slant character in **15/16** is not a field marker.

409. Relationship of Field Descriptors

Every field position has an assigned purpose that establishes the meaning of the data entered in that field. The first field position of a hypothetical linear set may address the place of departure, while the second identifies the intended destination. Without this pre-established positional meaning, the following set would be ambiguous:

```
TRIP/NEW YORK/LONDON//
```

The FUDs prescribed for the field positions establish the meaning assigned to that field position and, in most cases, the field contents provide sufficient prompting for the human reader to understand that meaning. There are cases, however, where an Information Exchange Requirement can only be satisfied by providing additional information to the human reader. The need for a field descriptor is determined during the design of the set format. Using the example shown above, the addition of field descriptors to the set enhances human understanding:

```
TRIP/FM:NEW YORK/TO:LONDON//
```

Field descriptors may be used to distinguish between two or more alternative field formats that can be used in a given field position. Field descriptors are a set attribute and are discussed in detail in Chapter 5.

410. Data Not Available

When the data needed to complete a field is not available, the hyphen (-) is used as the field content. If a composite FUD is specified, all elements of a composite field format must be known in order for it to be completed. As an example, FFIRN 2000 (Day and Time) is a composite field format composed of elemental FFIRNs 1000 (Day), 1001 (Hours), 1002 (Minutes) and 1003 (Time Zone). If the correct time zone is not known, FFIRN 2000 is not known, in its entirety, and the hyphen must be used as field content. In some cases, the data codes prescribed for a given FUD include an individual entry data code representation for **Unknown**, e.g., **UNK**. When such a data code is available, it must be used rather than the hyphen when the information to complete the field is not known. Further restrictions on the use of hyphen may be achieved by specifying expressions in the related formats in Parts II or III, in accordance with the rules given in Annex C.

a. Linear Sets

When the data needed to complete a field within a linear set is not available, the hyphen will be entered immediately following the associated field marker. Assuming sufficient space remains on the line, the hyphen will be followed immediately by either the next field in the set, e.g., **/-/XXX**, or the end-of-set marker when the affected field is the final field of the set, e.g., **/-//**. (Paragraph 313 discusses the use of continuation lines when sufficient space is not available.) Under circumstances discussed in Chapter 5, fields at the end of a linear set for which data are not available may be omitted. A completed linear set in which the data for the sixth field are not available is demonstrated by the following example:

```
REF/A/PURPLE/USS JOHN F KENNEDY/051400Z/005/-/JFK005//
```

When a field descriptor is prescribed by the set format and the information for the corresponding field is not available, the field descriptor will not be included in the completed field. Thus, **/PRF:-** is incorrect; use **/-** instead.

b. Columnar Sets

Hyphens are entered in the fields of columnar sets when corresponding data are not available . The hyphen will be left- or right-justified within the appropriate column, based on the justification prescribed for the FUD(s) associated with the column (see paragraph 503.f). For example:

```
5POL
/CODE  /POL TYPE   /CBM-OH   /DOS
/-     /DIESEL     /      800/   -//
```

In this case, the data required for the first and the fourth fields of the data line were not available. The hyphens in this example have been left- or right-justified, as specified by the set format for the FUDs associated with the first and last columns. When the hyphen is used as a field content, sufficient blank characters are added to maintain the fixed length of the field, as specified by the set format. Under circumstances described in Chapter 5, fields at the end of lines of a columnar set for which data are not available may be omitted.

411. Inappropriate Data

The hyphen is also used to complete a field when entry of a data code would be inappropriate. This situation arises when the applicability of a field is limited in some way. For example, a set designed for reporting information about a target area could include a field position for the radius of a circular target area followed by field positions for the length and width of a rectangular target area. Providing a data code for the radius establishes not only the size of the area but also that its shape is circular; information concerning length and width then becomes inappropriate, and the hyphen would be used as field content for the corresponding fields. The reverse would be true for a rectangular target area. Such fields are assigned conditional occurrence categories, as discussed in Chapter 5, and are explained in the related set format remarks. The rules for entering a hyphen in columnar fields are found in paragraph 410.

412. Selection of Data Codes

Sometimes it is necessary to place restrictions on the field content otherwise applicable for a field. This is called selection of field content and is achieved by using expressions specified in the related format remarks in Part II or III in accordance with the rules given in Annex C. Selection of field content based on message text format identifier is not allowed. In such a case new formats shall be defined.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5

SET LEVEL CONVENTIONS

501. Types of Sets

FORMETS employs three types of sets: linear, columnar and free text. Syntactically, they are common to the extent that each begins with a set format identifier that is followed by one or more fields and an end-of-set marker to terminate the set. Nevertheless, their structures differ significantly and reflect their intended usage in formatted message text.

502. Linear Set Structure

A linear set is constructed so that fields are presented in a linear manner. A linear set format may contain as many field positions as are necessary to satisfy the information exchange requirements addressed by the set. The following example of a linear set shows how fields follow one another on the same line:

```
MSGID/GREEN/CTG89.05/895001/MAR/REQ/1//
```

a. A linear set format identifier is composed of from three to eight alphabetic characters, preferably having mnemonic significance, that uniquely identify a set format. **MSGID** is the set format identifier for the above linear set.

b. The meaning of the contents of the fields of a linear set is established during the design of the set format by specifying the type of information to be provided in each field position. In the set demonstrated above, the six field positions were allocated for information concerning the message text format identifier, **GREEN**, the message originator, **CTG89.05**, the message serial number, **895001**, the month, **MAR**, the qualifier, **REQ** and the serial number of qualifier, **1**. This fixed relationship between field position and context meaning makes it essential that the order of fields be maintained in all applications of a given set format.

c. The final field format or GROUP OF FIELD FORMATS in a linear set format may be designated as being repeatable (explained further in paragraph 507).

503. Columnar Set Structure

Set formats designed for columnar presentation allow repetitive information to be arranged in vertically aligned columns under an appropriate column header. The fields of a columnar set often contain, as part of the field content, blank characters that are required to achieve the columnar presentation. The following set demonstrates the basic presentation of columnar set components:

```
2AMMOH
/CODE  /AMMO-NAME   /QTY-OH   /DOS
/AMMO50/50CAL       /     3000/   2
/HOW105/105MM       /      600/   2//
```

a. Set format identifiers for columnar sets begin with a numeric character, thereby providing distinction from linear and free text set format identifiers, followed by two to seven alphabetic characters that preferably have mnemonic significance. The set format identifier for a columnar set is entered alone on the first line of a columnar set. **2AMMOH** is the set format identifier for the above columnar set.

b. The second line of a columnar set comprises the HEADER LINE, which may occur only once. It contains column headers, blank characters and slants (/) as column delimiters, all of which are specified by the set format and must not exceed 69 characters. Column headers provide a cue to the human reader as to the type of information contained in each column, and are required, irrespective of the occurrence categories assigned to the corresponding fields by the columnar set format. In the example above, **CODE**, **AMMO-NAME**, **QTY-OH** and **DOS** are column headers. Column headers are composed of alphanumeric characters. All column headers are left-justified, irrespective of the justification of the field contents prescribed by the set format.

c. Columnar sets make use of the convention for grouping fields to allow for their repetition as a group (explained in paragraph 507). That is, after the header line each subsequent line of a columnar set is made up of a group of fields. All of the fields that are specified for the set must be included in this group. This group of fields shall not require more than 69 characters for all field markers and field contents (field descriptors - if used - data codes and additional blank characters for tabular representation as necessary). Each use of the group of fields is entered on a separate line, beginning at the first character position of the line. In the example above, the group of fields in the **2AMMOH** set is used twice.

d. The iterations of the group of fields in a columnar set contain the field contents that include the data codes associated with the FUDs assigned to the field positions by the set format. In the example above, **AMMO50**, **50CAL**, **3000** and **2** are data codes. In all but the first field of this example, blank characters are added as part of the field content to maintain the columnar alignment.

e. Each field will be fixed in length and contain sufficient character positions for the entry of either the field marker, the field descriptor (if used) and the longest code permitted by the assigned FUDs or the column delimiter and column header, whichever is greater. The starting character position for each field is, therefore, fixed and specified by the set format. In the example above, the number of character positions allocated to the fields are 7, 13, 10 and 5 respectively. In each case, the length of the permitted data codes exceeds the corresponding column header length. The start column character positions in this example are 1, 8, 21 and 31, respectively; column delimiters or field markers are entered in these positions. The following examples demonstrate the use of different types of data codes. In these three examples, the column widths are 21, 16, and 7 characters, respectively. Note how blank characters (represented by the symbol ρ) are added to pad the column header or data code to the width of the column, but that the parameters of the FUD itself are not affected by this padding.

```
(1)  FFIRN/FUDN                  1022/049    (1-20X)
     Character Position No:   12345678901234567890
     Column Header:           /SHIP-NAMEρρρρρρρρρρρ
     Field:                   /HMS ARK ROYALρρρρρρρ
```

```
(2)   FFIRN/FUDN                      2024/003     (15AN)
      Character Position No:     123456789012345
      Column Header:                  /LOCATIONρρρρρρρ
      Field:                     /301420N0804010W

(3)   FFIRN/FUDN                      4321/001     (2-6AN)
      Character Position No:     123456
      Column Header:                  /ELEVρρ
      Field:                     /ρ300FT
```

f. Data codes are left- or right-justified within the field according to specifications given in the set format in Part III for each FUD associated with a field position.

g. Field descriptors, which are explained in paragraph 506, are normally not used in columnar sets because column headers usually provide an adequate cue to the human reader as to the meaning of the data. Occasionally, it may be necessary to use a field descriptor to distinguish between alternative FUDs assigned to a single field position of the set format. In such fields the field descriptor will immediately precede the data code and the two will be left- or right-justified, as a pair, based on justification prescribed for the FUD in Part III.

h. The end-of-set marker (//) that terminates the set is positioned immediately after the final data code of the last iteration of the group of fields. However, it will be placed on the next line if the final data code extends beyond the 67th character position.

i. The structure of columnar sets is such that all columns must be contained within the length of one line, i.e., 69 character positions. Neither the header nor data lines may be continued on a second line. However, it is not always possible to provide for all types of information within this limitation. Two (or more) columnar sets must be designed in such cases to provide columnar presentation of information related to the entity or event being described.

j. The correlation of data between two or more related columnar sets can be achieved by the use of a DATA ENTRY IDENTIFICATION FIELD with column header DE as the first column. Unless otherwise specified within the message instructions, the set that first addresses an entity or event in the completed message will establish a numeric value to be reported under the DE column header. Data lines (groups of fields) in subsequent sets having the same numeric entry under the DE column header will be understood to pertain to the same entity or event. Correlation between sets can also be achieved by similar techniques when a common entry, such as mission number, would always be available to the message drafter. The following hypothetical examples demonstrate related columnar sets and the correlation of information contained therein:

```
(1)    6SHIP
       /DE/REF    /SID/QTY  /SHPTYP/HULL/TG
       /01/WSA304 /FRD/    1/TNK   /-   / 2
       /02/WSA304 /HOS/    1/KRESTA/-   / 1//
       7SHIP
       /DE/TIMPOS /SHIPLOC             /CRS/SPD
       /01/150930Z/320-COVIA-100       /330/   12KTS
       /02/150930Z/350-COVIA-130       /270/   10KTS//

(2)    6BASIC
       /MSNNO /REQNO /PRY/AMSN/TOT    /TFT    /ALRT/MSNLOC
       /WSA300/LNV542/1F /INT /161000Z/-      /-   /6TARGET
       /WSA301/LNV543/1C /CAS /161100Z/-      /-   /CHARLIE
       /WSA302/LNV544/2C /AEW /160800Z/161600Z/-   /ALPHA
       /WSA303/LNV546/2B /REC /160600Z/-      /-   /3240N10725W
       /WSA304/LNV549/1A /SCP /160500Z/-      /-   /340-ZZ-60
       /WSA305/LNV550/1D /AR  /160800Z/161100Z/-   /3000N10200W
       /WSA306/LNV551/1D /ESC /160700Z/161100Z/-   /3000N10000W//
       7CONTROL
       /REF   /C/CALLSIGN     /PRIFRQ  /SECFRQ  /REPIN
       /WSA300/F/GUNNER       /   259.3/   269.3/320000N1031600W
       /WSA301/I/SEAKING      /   342.1/   358.6/POINT BRAVO
       /WSA302/C/POPCORN      /   321.2/   343.0//
       6RDZ
       /REF   /L/RDZREF/RENDCS      /PRIFRQ  /SECFRQ
       /WSA300/N/WSA200/SEAWITCH 01 /   321.8/   315.5
       /WSA305/Y/WSA306/-           /   251.6/   358.4//
```

Through the use of the DE convention, the information provided in data lines of the 6SHIP and 7SHIP columnar sets in example (1) are identified as relating to the same entity or event. Similarly, WSA300 and the other mission numbers provided by the 6BASIC set in example (2) identify the entity or event about which information is provided in the 6BASIC, 7CONTROL, and 6RDZ sets.

504. Free Text Set Structure

Like other sets, free text sets begin with a set format identifier and are terminated with an end-of-set marker. A free text set contains a free text field of unrestricted length which is arranged in contiguous lines of up to 69 characters per line. The information conveyed in the free text field of free text sets usually contains natural language and may make use of all available characters (alphanumeric, colon and slant) except successive slant characters. There are four free text set formats: General Text, Amplification, Narrative, and Remarks (set format identifiers: GENTEXT, AMPN, NARR, and RMKS, respectively). They are the only set formats in which the free text field format is used. The use of the free text sets is described in paragraph 606.

a. The GENTEXT set contains two fields. The first field provides a formatted indication of the subject and is followed by a free text field with no prescribed length limitation.

b. The AMPN, NARR and RMKS sets contain only a single free text field. The text information in these sets provides supplemental information relating to:

- the preceding set, by means of AMPN,
- the preceding GROUP OF SETS, by means of NARR, and
- all of the sets in the message text, by means of RMKS.

505. Alternative Content Fields

Some types of information can be expressed in a variety of ways. Location, for example, can be stated as geographic coordinates, bearing and range, place name, etc. In such cases, the set format may specify two or more alternative field formats for the same field position of a linear or columnar set. These fields are known as ALTERNATIVE CONTENT FIELDS. For documentation in Part III, the alternative FUDs are listed with alphabetic identifiers (i.e., A, B, C, ... ZX, ZY, ZZ) in linear and columnar set formats. These identifiers are not transmitted as part of the message. The alternatives must be subject related, but need not have the same FUD structure; one FUD could require 2N, another 3-4A and a third 1-10X. In a linear set format, these structural differences cause subsequent fields, if any, to shift accordingly. In a hypothetical linear set SETID, where the second of three fields is an alternative content field allowing either 2N, 4A or 6X, the completed set could appear as:

```
SETID/FIELD 1/NN/FIELD 3//
SETID/FIELD 1/AAAA/FIELD 3//
SETID/FIELD 1/XXXXXX/FIELD 3//
```

Use of alternative contents of differing FUD structures has greater consequences in columnar sets, for the different lengths must be taken into account when determining the width of the related column and positioning (justification) of data within the column. Using the same alternatives and assuming a column header of three alphabetic characters (XYZ), the column width becomes seven character positions (including the column delimiter or field marker), as demonstrated below. Note that the arrangement of data and the number of required blank characters (represented here by ρ) vary between alternatives.

```
Column Header:   /XYZρρρ
Alternative 1: /ρρρρNN
Alternative 2: /AAAAρρ
Alternative 3: /XXXXXX
```

If the field contents allowed by the various alternatives are not distinguishable from one another, the associated set format must provide a means to identify the alternative chosen. The means to accomplish this identification is through the use of field descriptors.

506. Field Descriptors

A field descriptor is a word, an abbreviation or an acronym. It comprises from one to eight alphabetic or numeric characters followed by a colon (:) to separate it from the data code that follows. The field descriptor is used if there is an operational requirement to cue the human to the meaning of the field content or to identify the alternative used in an alternative content field. The use of a field descriptor in a particular set is as prescribed by the set format in Part III. The need for a field descriptor is determined during the design of the set format. It should be remembered, however, that context meaning is provided by the association between field position and assigned FUD, not by the use of field descriptors. Field descriptors shall be prescribed for alternatives in a field position where alternative contents cannot be distinguished by the number and type of characters or by data codes found in Part IV. If the number of alternative contents that cannot be distinguished is N, then the minimum number of field descriptors prescribed shall be N - 1. Field descriptors are not permitted in free text set formats.

507. Field Repetition

There is often a need for repetition of certain types of information within a set. To support this need, one or more fields of a set may be repeated as described in the subparagraphs below, but only when so designated by the set format structure in Part III. There is no limit to the number of repetitions unless this is specifically limited in Parts II or III, in accordance with expressions described in Annex C. There is no restriction on the selection of an alternative in repeated alternative content fields unless that restriction is accomplished in accordance with expressions described in Annex C.

a. Repetition Within Linear Set

In the linear set, the designated final field, or group of contiguous fields that includes the final field, may be repeated. The following demonstrates how the final field of a four-field set would be repeated:

```
SETID/FIELD 1/FIELD 2/FIELD 3/FIELD 4/FIELD 4/FIELD 4//
```

Repetition of the final fields of a set as a group of fields must be accomplished in a way that allows context meaning of repeated field positions to be maintained. Therefore, when a group of fields is designated for repetition, it must be repeated as a unit. In the following example, Fields 2 and 3 have been designated as a repeatable group of fields, which is repeated once. The SHIPS set is composed of three fields, the final two of which are repeatable as a group of fields. The set could be completed as follows:

```
SHIPS/US/BB/USS NEW JERSEY/CVN/USS NIMITZ//
```

In this case, US is the field content for field 1, BB is the field content for the initial field 2, while CVN has been entered for its second use.

In repeating the final fields of a set as a group of fields, the order of fields within the group must be maintained, even if some of the data were not available. Using the example above, if the information for Field 2 were not available, the hyphen (-) must be used as field content, which can be demonstrated as follows:

```
SHIPS/US/BB/USS NEW JERSEY/-/USS NIMITZ//
```

b. Repetition Within Columnar Set

All the fields of a columnar set comprise a repeatable group of fields, as explained in paragraph 503. In columnar sets, each use of the group of fields is always entered on a new line in order to maintain the desired tabular arrangement. In the following example, the first use of the group of fields begins with the data code AMMO50, the second with HOW105, and so on.

```
2AMMOH
/CODE  /AMMO-NAME   /QTY-OH   /DOS
/AMMO50/50CAL       /     3000/   2
/HOW105/105MM       /      700/   2
/AMMO09/9MM         /     2000/   -//
```

c. Repetition Within Free Text Sets

Repetition of fields within free text sets is not permitted.

508. Occurrence Categories of Fields

Every field position in a set format is assigned an occurrence category based on the necessity and structural relevance of the information to the purpose and the use of the set. These occurrence categories remain constant regardless of the set usage in different message text formats. As explained in paragraph 316, the occurrence categories are Mandatory, Operationally Determined and Conditional. The following occurrence category rules apply to fields of a set.

a. A mandatory field position must be completed for every use of the set. (Note: Entry of a hyphen (-) to represent a lack of data constitutes field completion.)

b. An operationally determined field is to be completed whenever it is appropriate to do so. Where data for any reason cannot be entered, the field may be omitted, unless it is followed by one which requires completion, in which case a hyphen (-) is entered.

c. A conditional field position has a condition governing its use in the set. That condition must be explicitly stated in Part II or Part III depending on the containing format which allows the internal verification. If other fields following such a field in a set are used, and the condition prohibits the use of the field, a hyphen(-) must be entered.

509. Truncation of Sets

Truncation of a set by the omission of one or more field positions is permitted as follows:

a. Unused fields at the end of a linear set may be omitted:

```
      (1)
Field Position No:         1 2         3       4       5  6 7
Category:                  M M         M       M       M  O O
Data Line:              REF/A/NAVSITREP/SACLANT/202200Z/01/-/703//
```

```
      (2)
Field Position No:         1 2         3       4       5
Category:                  M M         M       M       M
Data Line:              REF/B/NAVSITREP/SACLANT/212200Z/01//
```

In the example (1), all of the fields in the set are completed since the information for the last field, which is an operationally determined field, is available; truncation is not possible. Example (2) shows another use of the same set in which the last two operationally determined fields are omitted; the set has been truncated by placing the end-of-set marker after the fifth field.

The following examples demonstrate the treatment of a linear set of a message text format when none of the data to complete its fields are available. The left column depicts hypothetical field occurrence categories for the respective set formats as shown in message text formats; the right column shows how the set should be completed.

DRCTN/M/O/O//	DRCTN/-//
OPSTATUS/M/M/O//	OPSTATUS/-/-//
ORGSTAT/O/O/O/O/O/O/O//	ORGSTAT//

Fields with an occurrence category of C are treated exactly as those with an occurrence category of O when the conditions do not require or prohibit the use of the fields.

b. The following example demonstrates the application of truncation conventions in a columnar set. It also demonstrates how these conventions apply to fields having conditional occurrence categories. In this case, assume that the conditions for fields four and five were not met (these fields are therefore treated as operationally determined). Assume also that information is available for the fourth field of Data Line 1 and the fifth field of Data Line 2, but not available for other iterations of these fields. Note that Data Lines 1, 3, and 4 have been truncated and that the end-of-set marker is placed after the last entry in the final data line, in this case after the data entered in the third field.

```
Field Position
 (Column) No:     1      2            3      4      5
Category:         M      M            M      C      C
Set Format
 Identifier:      1FWPNS
Header Line:      /CODE  /NAME        /QTY-OH/QTYSVC/QTYTOE
Data Line 1:      /HOW105/105MM       /    15/    13
Data Line 2:      /MACH50/50 CAL MACH /    37/     -/    60
Data Line 3:      /-     /81 MM MORTAR/    18
Data Line 4:      /MORT42/4.2IN MORTAR/     -//
```

In the event that no data is entered in any field of a columnar set the set is truncated. That is it consists only of a set identifier followed by the end-of-set marker on the same line.

510. Empty Sets

An empty set, that is, a set in which none of the fields are used and which, therefore, consists of only a set identifier and an end-of-set marker, must be omitted unless its use in the message or its containing segment is governed by a mandatory occurrence category or is required by a condition.

This page is intentionally blank.

CHAPTER 6

MESSAGE TEXT LEVEL CONVENTIONS

601. Message Text Format

The message text format identifies the set formats and segments that may be used in a formatted message and the order in which they must be arranged.
A unique message text format identifier, which may be composed of from one to twenty characters (except slant and colon), is assigned to each message text format and appears in the first field of the MSGID set of a completed message. The field content (i.e., the MESSAGE TEXT FORMAT IDENTIFIER) therefore serves as a key to the structure and information content of the message text format. The message text format also specifies additional formatting conventions that are, or may be, applied at the message level.

602. Order of Set Formats

The overall composition of the message text was discussed in Chapter 3. As explained there, message text formats document only the main text structure. The set formats with identifiers EXER, OPER, MSGID, and REF occupy, in this sequence, the first set positions in each MTF to the extent that they are specified. Of these set format identifiers only MSGID must be specified in each MTF. Care must be exercised in designing MTFs to avoid introduction of an ambiguous set order which may, in a completed message, result from a particular combination of specified set occurrence and repetition. Sets follow one another on succeeding lines. For example, if the REF set is not required in a given message, the next set required by the message text format will begin on the line immediately following the MSGID set. The order of sets within a completed message must conform to the sequence specified in the message text format. The main message text may be interrupted, where called for by applicable communications procedures, to enable insertion and transmission of information concerning page numbering, message sectioning or similar communications-system-related requirements.

603. Individual Set Repetition

During the design of an MTF, individual sets may be designated as repeatable in the message text format in Part II. There is no limit to the number of times a set may be repeated unless specifically limited in Part II, in accordance with expressions described in Annex C. Repeated sets follow immediately after the original set on the next line of the formatted message. For example:

```
CIRC/3500N06000W/150NM//
CIRC/3810N06500W/275NM//
```

Certain sets will not be designated as being repeatable because of their structural significance or the nature of their information content. These are the EXER, OPER and MSGID sets, free text sets and the initial set of a segment.

604. Segmentation

Contiguous sets that are related by content and are to be allowed to repeat as a group are designated as a segment during the design of a message text format. Segments may be nested within segments as necessary. There is no limit to the number of times a segment may be repeated unless specifically limited in Part II, in accordance with expressions described in Annex C. By specifying, in Part II, a limit of only one occurrence, a non-repeatable segment can, in effect, be designed. When repeated, a segment occurrence will follow immediately after the original segment occurrence. The first set of a segment is designated by a segment occurrence category (M, O, or C) to the left of that first set. Subsequent sets in the segment are designated by square brackets ([) to the left of the set name. The following is an example of a mandatory segment composed of sets EPLACE, GRAPH, OPSTATUS, SOURCE and TIME.

```
M     EPLACE
[     GRAPH
[     OPSTATUS
[     SOURCE
[     TIME
      (End of EPLACE segment)
```

Within the following mandatory segment, sets LOCATION, OPSTATUS and TIME make up a conditional NESTED SEGMENT while sets TERMINAL, LINKSTAT and TIME, and sets COURIER and TIME form two other nested segments whose use is operationally determined:

```
M      ORGID
[      ORGSTAT
[C     LOCATION
[[     OPSTATUS
[[     TIME
[           (End of LOCATION segment)
[O     TERMINAL
[[     LINKSTAT
[[     TIME
[           (End of TERMINAL segment)
[O     COURIER
[[     TIME
[           (End of COURIER segment)
       (End of ORGID segment)
```

a. The initial set of a segment should provide information as to the overall subject of all sets comprised by the segment. The initial set of a segment or nested segment may be selected from two or more mutually exclusive sets. Such mutually exclusive initial sets are also called alternative initial sets.

b. Initial sets cannot be specified as being individually repeatable. Other sets within segments may be specified as individually repeatable within the containing segment as required.

c. The GENTEXT set is the only free text set that may be used as an initial set of a segment.

d. Care must be exercised in designing segments to avoid introduction of an ambiguous set order in a completed message text. Examples of structures that could lead to ambiguity include (and are demonstrated below):

(1) Using the same set as the initial set of two segments (on the same level of nesting) that are not separated by at least one mandatory set.

(2) Using the same set as the initial set of a segment and any nested segment included therein.

(3) Using the same set as the initial set of a nested segment and elsewhere within the parent segment.

(4) Assigning occurrence categories that introduce potential ambiguities arising from non-use of conditional or operationally determined sets.

```
Example      Example      Example      Example
  (1)          (2)          (3)          (4)

 M A          M  A         M  A         M A (M)
 [ B          [  B         [  B         [ B (C)
 [ C          [M A         [M C         [ C (O)
 M A          [[ C         [[ D           C (M)
 [ D          [[ D         [  C
 [ F          [  E         [  E
```

605. Occurrence Categories of Segments and Sets

Every segment and set in a given message text format is assigned an occurrence category based on the importance and the structural and operational relevance of the information to the purpose being served by its next higher level containing format (see paragraph 316). The occurrence category of a nested segment refers to its use within the containing segment. The following special conventions apply to occurrence categories of the set or segment as indicated.

a. Occurrence Category Conventions

(1) Any set whose occurrence category is mandatory must be included whenever its containing segment is used. If the set is not in a segment, then it must be included in the message.

(2) Any segment that is not nested within another segment, and whose occurrence category is mandatory, must be included in the message.

(3) Any nested segment whose occurrence category is mandatory must be included whenever the containing segment is used.

(4) The occurrence category of the initial set of every segment is mandatory unless there are alternative initial sets. In this case all of the alternatives shall be mutually exclusive and one of them (only) shall be used. If, for example, the selection of the alternative is to be operationally determined, one of the alternatives will have an occurrence category of operationally determined while all of the other alternatives have an occurrence category of conditional.

b. Occurrence Related Rules Applications

The following example illustrates a hypothetical message text format containing set and segment occurrence categories:

Segment Occurrence Categories	Set Occurrence Categories	Set ID
	M	Set A
C	[M]	Set B
[	M	Set C
[M	[O]	Set D
[[	[C]	Set E
[[	O	Set F
[	O	Set G
	O	Set H

This example illustrates the convention for specifying segments and the associated occurrence categories. It also shows how occurrence categories of the individual sets are depicted.

The start of a segment is indicated by its occurrence category (M, C, or O) in the Segment Occurrence Categories column on the same line as the segment's initial set. The remaining sets in the segment are indicated by a square bracket ([) on subsequent lines in the same column. The segment ends when the square brackets cease. In the example, the first level segment has an occurrence category of conditional and is made up of sets B through G.

The example also shows a second level nested segment which is made up of sets D through F. It has an occurrence category of mandatory, meaning that it, the nested segment, must be used each time that its containing segment (sets B through G) is used.

In the above example the Set Occurrence Categories column shows the occurrence category of each set. This column also indicates the initial set(s) of segments by enclosing their occurrence category indications in square brackets ([]). Note that the domain over which the set occurrence category applies is shown by the information in the Segment Occurrence Categories column. The statements below, for each set in the example, show how this is done.

Set A is mandatory. It must be included each time the message is used.

Set B is mandatory. It must be included each time the segment is used within the message. It is the initial set of a segment.

Set C is mandatory. It must be included each time the segment is used within the message.

Set D is operationally determined. It is used each time the nested segment is used if the operational situation requires it. It is an initial set of the nested segment, as indicated by the square brackets enclosing its occurrence category indicator.

Set E is conditional. There is a statement in the remarks section of Part II indicating the structural conditions under which this set is required or prohibited whenever the nested segment is used. Since this set is an alternative initial set of the nested segment, as indicated by the square brackets enclosing its occurrence category indicator and that of its immediately preceding set D, the conditional statement in Part II will be that set E is required if set D is not used and prohibited if set D is used. Note that these conditions apply only over the domain of each occurrence of the nested segment. That is, whenever the nested segment is used, one or the other (not both) of sets D and E must be used.

Set F is operationally determined. It is used each time the nested segment is used and the operational situation requires it.

Set G is operationally determined. It is used each time the first level segment is used and the operational situation requires it.

Set H is operationally determined. It is used if the operational situation requires it.

The following are examples of correct set sequences, assuming that associated conditions, whatever they may be, are met:

A,B,C,D,F,E,F,B,C,D,G

A,H

A,B,C,E,B,C,D

The following are examples of incorrect set sequences:

A,B,D	Set C was omitted while its containing segment is used.
A,D,F,G	The nested segment is used but its containing segment is not.
A,B,C,G	The segment is used but its mandatory nested segment is not.

606. Free Text Sets

There are two categories of free text set formats. One category is for use by the MTF designer. This is called "scheduled" use. The other category is for the use by the message drafter. This is called the "unscheduled" use. Scheduled free text sets are documented in the Message Text Format. Unscheduled free text sets are used at the discretion of the message drafter in accordance with the rules listed in paragraph 606.b.

a. Scheduled Use

The GENTEXT set format is the only free text set permitted for scheduled use.

(1) The GENTEXT set may be used to provide free text information in amplification of the preceding set, an immediately preceding group of sets, or the whole message. It can also be used to provide free text information not specifically related to other sets.

(2) The GENTEXT set format has two fields. The first field provides a formatted indication of the subject which is addressed in the following free text field and, where appropriate, its relationship to preceding sets.

(3) The GENTEXT set format may not be designated as repeatable. However, it may be specified for use in successive set positions with different subject indicators in the first field.

(4) The GENTEXT set format may be used as the first set format of a segment.

(5) The use of the GENTEXT set format shall be minimized by the MTF designer. Only when the information cannot be designed into formatted sets should the GENTEXT set format be used.

b. Unscheduled Use

The AMPN, NARR, and RMKS are free text sets for unscheduled use.

(1) The AMPN set is used to provide an explanation or additional information concerning only the preceding set.

(2) The NARR set is used to provide an explanation or additional information concerning an immediately preceding group of two or more contiguous sets.

(3) The RMKS set is used to provide an explanation or additional information applicable to the whole message. When used, it will always be the last set of the main text of the message.

(4) AMPN, NARR, and RMKS free text sets are not repeatable. A formatted message may not contain successive AMPN or NARR sets, and only one RMKS set may be used in a given message. Depending on the structural relationships of required free text information, it is possible for an AMPN set to be immediately followed by a NARR or the RMKS set, or for a NARR set to be immediately followed by the RMKS set. The AMPN set cannot immediately follow the NARR set.

ANNEX A

SYNTAX OF THE NATO MESSAGE TEXT FORMATTING SYSTEM

1. General

This annex provides a formal and unambiguous definition of FORMETS syntax.

In drafting messages and preparing programs for computerized analysis of formatted messages the syntax is the binding set of general rules governing the construction of terms in the formatted text. For a formatted message to be legal appropriate definitions must also have been entered in the catalogues of Parts II to IV.

In developing formats the syntax specifies the structures which can be defined through message, set and field formats.

2. Syntax Notation

To describe the syntactical structure, a subset of the commonly used Backus Naur Form (BNF) employing the descriptors "expression", "term" and "operator" is used.

c. "Expression" refers to a combination of terms and operators. An expression making up a complete syntactical definition is called a production.

d. "Term" is a word, abbreviation or a group of words and abbreviations from natural language, e.g. formatted set, set or end of set. It is placed between angular brackets, e.g., <term>. It may be replaced, where appropriate, by a specific representation by enclosing the representation in double quotes, e.g., "EXER", "A", "//".

e. "Operator" defines syntactical relations between expressions and/or terms. The following operators are used:

Operator	**Meaning**	**Explanation**
::=	definition operator	The term to the left of the operator is defined by the expression to the right.
[]	optional items	The term(s) or expression enclosed by the brackets may occur once or not at all.
{}(n,m)	repeatable	The term(s) or expression enclosed by curly brackets shall be repeated so that they occur at least n times and maximum m times. If no value is given for m no maximum value for the number of occurrences is defined. If the curly brackets are not followed by round parenthesis with n and m, the enclosed terms shall occur at least once and no maximum value for the number of occurrences is defined. When the repeatable brackets are enclosed by optional items brackets (e.g. [{<a>}]) the items may occur any number of times - including zero times.
|	alternative operator	All terms to the left of the operator up to the next operator is one alternative and all terms to the right down to the next operator or end of expression is another alternative. One and only one of the alternative term(s) or expressions can occur. This operator has a lower precedence than the other operators such that for instance <e>::=<a><b>|<c><d> means that there are two alternatives for <e>, either <a><b> or <c><d>. Similarly <f>::=<a>{<b>|<c>} means that <f> is made up of one <a> followed by a sequence of <b>s and/or <c>s such as <a><c><c><b><c>.

3. Message Text Syntax.

The productions defining the message text syntax for FORMETS are given in the following table. A number of general rules and conditions apply.

- The actual structure of specific MTFs, set formats and field formats are specified in ADatP-3, Parts II - IV.

- When a message is composed the full line length of 69 characters shall be used as far as possible, subject to the rules governing where a new line can occur.

- Definitions complementing the formal syntax and comments are added enclosed by the character * and written in italics e.g. **this is a comment**.

- Each production is labelled by a number for ease of reference. The number is not part of the syntax.

- Productions marked with * in front of the reference number are not needed for the BNF definition itself but are added to define terms used in the main text of this document.

0 <text>::=[<introductory text>]<main message text>[<closing text>]

FORMETS only covers the specification of main text. This production is accordingly not part of FORMETS but is included to show the relation between the main text and other components which might be of relevance. For the same reason the following productions will not provide definitions for <introductory text> and <closing text>.

1 <main message text>::=<first set group>[{<set group>}][<remark>]

2 <first set group>::=<EXER-OPER set group><MSGID set group>[<narrativ
e>][{REF set group}]|<MSGID set group>[{REF set group}]

3 <EXER-OPER set group>::="EXER"<exercise identification set fields><en
d of set>[<amplification>]|"OPER"<operation codeword set fields><
end of set>[<amplification>]

<exercise identification set fields> and <operation codeword set fields> are sequences of <linear field>s as defined in Part III.

4 <MSGID set group>::="MSGID"<field marker><message text format identifier><additional message identifier set fields><end of set>[<amplification>]

<additional message identifier set fields> are a sequence of <linear field>s as defined in Part III.

5 <REF set group>::="REF"<reference set fields><end of set>[<amplification>][<narrative>]

<reference set fields> are a sequence of <linear field>s as defined in Part III.

6 <message text format identifier>::=<extended character>[{<alphanumeric character>}(0,18)<extended character>]

7 <set group>::=<amplifiable set>[amplification>][<narrative>]

8 <amplifiable set>::=<linear set>|<columnar set>|<general text>

9 <amplification>::="AMPN"<free text field><end of set>

10 <narrative>::="NARR"<free text field><end of set>

11 <remark>::="RMKS"<free text field><end of set>

12 <general text>::="GENTEXT"[<new-line>]<subject indication field>[<new-line>]<free text field><end of set>

13 <subject indication field>::=<field marker><data code>

*14 <set>::=<linear set>|<columnar set>|<free text set>

*15 <free text set>::=<amplification>|<narrative>|<remark>|<general text>

*16 <segment>::=<set group>[{<set group>|<segment>}]

17 <free text field>::=<field marker>[{<not-slant character>["/"]}]<not-slant character>

18 <linear set>::=<linear set format identifier>[{[<new-line>]<linear field>}][<new-line>]<end of set>

19 <linear field>::=<field marker><linear field content>

<linear field> maximum length is 69.

20 <linear field content>::=<data code>|<field descriptor><not-hyphen charact

er>[[{<alphanumeric character>}]<extended character>]

21 <data code>::=<extended character>[[{<alphanumeric character>}]<extended character>]

22 <field descriptor>::={<alphabetic character>|<numeric character>}(1,8)":"

23 <linear set format identifier>::={<alphabetic character>}(3,8)

24 <columnar set>::=<columnar set format identifier>[<header line>{<new-line>{<columnar field>}}[<new-line>]]<end of set>

The <header line> shall contain as many <column header>s as the number of <columnar field>s specified in the set format. The length of a <column header> shall be equal to the length of the corresponding <columnar field>. Total length of <columnar field>s shall not exceed 69 characters.

25 <columnar set format identifier>::=<numeric character>{<alphabetic character>}(2,7)

26 <header line>::=<new-line>{"/"<column header>[{<blank character>}]}

27 <column header>::=<extended character>[[{<alphanumeric character>}]<extended character>]

28 <columnar field>::=<field marker><columnar field content>

29 <columnar field content>::=[{<blank character>}]<linear field content>|<linear field content>{<blank character>}

*30 <field content>::=<linear field content>|<columnar field content>

*31 <formatted field>::=<linear field>|<columnar field>|<subject indication field>

32 <field marker>::="/"

33 <end of set>::="//"<new-line>

34 <new-line>::=

**<new-line> represents a code sequence which causes whatever follows it to start at the beginning of the next line in a given processing environment.*
*<new-line> is not a part of the ADatP-3 character set but is included in the productions purely to define where a new line must or may occur within a message as it will appear to a user.**

35 <blank character>::=" "

The character causing a blank (space) to be printed or displayed.

36 <numeric character>::="0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

37 <alphabetic character>::="A"|"B"|"C"|....|"Y"|"Z"

38 <alphanumeric character>::=<alphabetic character>|<numeric character>|<
special character>|<blank character>

39 <special character>::="."|","|"-"|"("|")"|"?"

40 <extended character>::="-"|<not-hyphen character>

41 <not-hyphen character>::="."|","|"("|")"|"?"|<alphabetic character>|<numeric
character>

42 <not-slant character>::=":"|<new-line>|<alphanumeric character>

*43 <available character>::="/"|<not-slant character>

This page is intentionally blank.

ANNEX B

FIELD FORMAT SPECIFICATIONS

THIS PAGE IS INTENTIONALLY BLANK

1. General

This Annex provides further instructions as to how field formats shall be documented in Part IV. Strict compliance with these rules allows distribution of unambiguous and computer processable specifications.

2. Field Format Specification

A field format has been defined as a specification contained in Part IV. That specification includes of the following elements:

a. Field Format Index Reference Number (FFIRN)

The FFIRN comprises four digits, 0001 through 9999, that uniquely identify a field format.

b. Field Format Name

The FF name is an identifying name that is unique among all FFs.

c. Field Format Structure

The FIELD FORMAT STRUCTURE is a character range that establishes the minimum and maximum number of characters permitted for data codes associated with the FF and the type of characters permitted for these codes.

d. Field Format Definition

The FF definition is a unique definition of the concept represented by the FF's set of data items if it is an elemental FF, or the set of concatenated data item strings if it is a composite FF.

e. Field Format Data Items and Data Codes

All of the data items (and their codes) that are to be associated with an elemental FF are listed as part of the FF specification. Each data item must be unique within the FUD. Similarly, each data code must be unique within the FUD. This list of data items and data codes is present only in the specification of an elemental FF. See paragraph 5 for further definitions of data item and data code specifications.

f. Component Field Use Designator

The FFIRN and FUDN (see paragraph 3.a below) of each component FUD is listed in the order in which their codes are to be concatenated. This list of FFIRN/FUDNs is present only in the specification of composite FFs.

g. Field Format Remark

The FF remarks include clarifying information that does not properly fit elsewhere in the specification. If the FF specification has classified information, the remarks shall identify the source of the classified information.

h. Field Use Designator Specification.

See paragraph 3 below.

3. Field Use Designator Specification

One or more FUDs must be specified in each FF specification. The FUD specification is defined in the following paragraphs.

a. Field Use Designator Number

The FUDN is a three digit number that is unique among the FUDs of the parent FF. A FF shall always have a FUD numbered 001.

b. Field Use Designator Name

The FUD name is a name that is unique among the FUDs of the parent FF.

c. Field Use Designator Data Items and Data Codes

All of the data items (and their codes) that are to be associated with the FUD are listed as part of the FUD specification. These data items and their associated codes must be selected from the data items and their associated codes that have been specified for the parent FF. If the only data item/data code type selected for association with a FUD is an instructive entry, that entry must specify literal. A FUD shall not be assigned both a literal data item entry and other data item entries of either individual or range entry types. The association of each data item with a data code is established as part of the FF and may not be altered in the FUD specification. This list of data items and data codes is not present as part of the specification of composite FUDs.

d. Field Use Designator Structure

The FUD structure indicates the structure of all of the data codes associated with a FUD by designating the number and permitted type(s) of characters.

e. Field Use Designator Explanation

An explanation of its particular use is associated with each FUD.

4. Character-Ambiguous Composite Field Format

A character-ambiguous composite FF is one that has adjacent component FUDs whose data codes are such that it may be impossible to determine which character belongs to which data code. Such a FF is an illegally designed FF and is not to be accepted in the standard. A character-ambiguous composite FF is possible only if its use results in a variable-length field. An example of a character-ambiguous composite FF is one with two adjacent component FUDs that have data codes ranging from 0 through 99. No way exists to determine whether, for example, the composite data code string 315 represents component data codes of 3 and 15 or of 31 and 5. Note that this would not be character-ambiguous if the range for each component FUD were given as 00 through 99. In this latter case the composite field would have a fixed length and the composite data code 315 would itself be illegal. That is, the legal code would have to be 0315 or 3105. In either case, the number of characters provides the key to determining the data code for each component. To ensure that precise boundaries between component FUD codes used in variable-length composite FFs can be determined, unambiguous separators may be used. Such a separator can be provided by a fixed-length component FUD to which only one individual entry data item and data code has been assigned. That data code (normally only one character such as a blank or hyphen) must be a code which cannot be used as part of the data code for either of the adjacent component FUDs.

5. Data Item and Data Code Entry

There are three data item and data code entry types: individual, range and instructive. Any number of entries can be specified, and the entries can be mixed arbitrarily within the Field Format. Validating a value against a FUD implies evaluating its individual Data Code entries. If the FUD contains an instructive entry having the word LITERAL as the first word (see paragraph c below), any appropriate value (see paragraph 401.c) is legal - otherwise the value must match one of the individual or range entries.The specification for the three types are explained below and also expressed formally in a BNF notation in Appendix 1.

a. Individual Entry

An individual entry is recognized by the fact that the first character is not an opening left square bracket ([).

b. Range Entry

A range entry is a short-hand notation for defining a set of legal values. It is recognized by the fact that it starts with a single opening left square bracket ([). The entry is terminated with a closing right square bracket (]). There are three kinds of range entries as explained below.

(1) Integer Range. If all values in a range are integers then the range entry consists of the bracketed statement [a THROUGH b] where a is the initial value in the range, b is the final value in the range, and it is understood that integer incrementation in the range is by 1. The initial data item value is the algebraically smallest value and the final data item value is the algebraically largest value. The data code sequence in a range shall be in a one-to-one correspondence with the associated data item sequence. Examples are as follows:

(a) [-100 THROUGH 100]

(b) [-99 THROUGH -1]

(c) [0 THROUGH 99]

(d) [00 THROUGH 99]

(e) [10 THROUGH 12]

(f) [1 THROUGH 1000000]

(2) Range Containing Decimal Point Value. If some or all values in a range are decimal point values (i.e., values which contain a decimal point), the range entry consists of the applicable bracketed statement followed by a parenthetical statement identifying the decimal place or decimal places involved. Decimal places are counted as digits after the decimal point, i.e. '.0' will count as 1 (one) digit. Specifying 0 (zero) decimal places means that the decimal point must not be present. The parenthetical statement can take three possible forms.

(a) (c DECIMAL PLACES) indicates that all values in the range always contain the same number of decimal places: c (i.e., the decimal point does not change position for different values within the range).

(b) (c TO d DECIMAL PLACES) indicates that all decimal places ranging from c through d occur where c is the minimum number of decimal places and d is the maximum.

(c) (c OR d DECIMAL PLACES) indicates that only two numbers of decimal places occur where c is the smaller number, d is the larger, and c and d are not necessarily adjacent numbers.

In forms (b) and (c), the parameter c can take on a value of zero indicating that one or more values in the range are not decimal point values. Form (a) does not permit zero. Examples are given below, each with explanatory comments. In each example, the accompanying explanation defines the set of integer values, if present, and what is the set of decimal point values that logically must be inferred from the range statement and parenthetical statement. Except where an explanation is absolutely unnecessary, a range entry containing decimal point values should be accompanied by a description of range values in the Explanation column to serve as a backup to the entry.

(d) [.0 THROUGH .9]
(1 DECIMAL PLACES)

Here the values are decimal point values of .0, .1, .2, etc, .8, .9. No backup description of values in the Explanation column should be necessary. In contrast, the examples below would require a backup description in the Explanation column.

(e) [0 THROUGH 9.9]
(0 TO 1 DECIMAL PLACES)

The values are the integers 0 through 9, and decimal point values. The decimal point values consist of all possible combinations of two or fewer significant digits and a significant decimal point. The above wording constitutes an adequate backup description of values. Note that the maximum field size is three characters to allow for certain decimal point values and could theoretically contain integers up to 999, but that 9 is the maximum integer permitted by this specification.

(f) [.000 THROUGH 9.999]
(3 DECIMAL PLACES)

Here the values are in the range from .000 through 9.999 in increments of .001.

(g) [0 THROUGH 9.999]
(0 OR 3 DECIMAL PLACES)

Here the values are the integers 0 through 9 and decimal point values. The decimal point values consist of all values in the range of .001 through 9.999 in increments of .001 containing a significant decimal point.

(h) [0 THROUGH 9.999]
(0 TO 3 DECIMAL PLACES)

Here the values are also the integers 0 through 9 and decimal point values, but the set of decimal point values is different from the example (g) above. Here the decimal point values are all possible combinations of three or fewer significant digits and a significant decimal point in the range .001 through 9.999.

(i) [0 THROUGH 999.9]
(0 TO 4 DECIMAL PLACES)

Here the values are the integers 0 through 999, and the decimal point values. Decimal point values are all combinations of four or fewer significant digits and a significant decimal point in the range .0001 through 999.9.

(3) Alphabetic or Alphanumeric Range. If all values in a range are alphabetic or alphanumeric, then the range entry consists of the bracketed statement [d, e, THROUGH f] where d is the initial value in the range, e is the second value, and f is the final value. All values d, e and f must have the same length. The inclusion of e provides the necessary information on the incrementation of values within the range.

A general semantical meaning of the syntax - using two letter values - is seen as:

[p a1 a2 s, p a3 a4 s THROUGH p a5 a6 s]

where a1, a2,..., a6 each identify a single character e.g. a1 = A (not AB) and a1a2 = AB. p/s identifies a constant prefix/suffix of any length (possibly zero), see example 5.b.(3)(m).

In what follows constant prefixes/suffixes are ignored, because they do not influence the range of legal values.

The a1 and a2 define the value of the character position to be used when starting the increment of the position; e.g.
[AB, AC THROUGH ZZ] specifies that values cycle from AB through AZ, then BB through BZ, then CB through CZ, etc. that is, in this example, the a2 position will always start with the value B.

The a3 and a4 define the second legal value of the sequence.
The difference between a1 and a3 defines the increment to be

performed on each step for the first character of the value.

The difference between a2 and a4 defines the increment to be performed on each step for the second character of the value.
If one of the increment values is 0 (zero), this implicitly means an increment of 1 (one), when the neighbouring position reaches its maximum, i.e. the carry is always 1 (one).
E.g. [AA, AB THROUGH ZZ] has the increment value (0, 1), i.e. when cycling from AA through AZ, the next value will be BA, as the a1 field is incremented by 1 (one) when a2 reaches its maximum.
All increments must have the same value or 0 (zero). This means increment values like (0, 1), (1, 0), (1, 1) and (2, 2) are allowed, but (1, 2) and (3, 2) are not allowed.

Increment values must appear at one of the ends of the range; e.g.
[AAA, AAB THROUGH ZZZ] with increment value (0, 0, 1),
[AAA, ABB THROUGH ZZZ] with increment value (0, 1, 1),
[AAA, BBA THROUGH ZZZ] with increment value (1, 1, 0) and
[AAA, BBB THROUGH ZZZ] with increment value (1, 1, 1) are all legal, whereas [AAA, ABA THROUGH ZZZ] with increment value (0, 1, 0) is not allowed (see examples b.(3)(j), b.(3)(k), b.(3)(l) below).

The a5 and a6 characters define the last value of the character positions to be used when incrementing; e.g. [AA, AB THROUGH BZ] implies values cycle from AA through AZ then BA through BZ and no more, i.e. in this example the first character position can have values either A or B and no other values.
The last value should always be reached; e.g. statements like [AA, AC THROUGH ZZ] are illegal, because with increment value (0, 2) a6 would never reach the value Z, as Y would be the last value.

Examples are given below with explanatory comments. Unless otherwise indicated, each alphabetic or numeric character cycles normally through the complete range of characters from the least to the most significant character. . When mixing numbers and letters in the same position it is predefined that the value of digits is less than the value of letters (see example b.(3)(i)).An appropriate backup comment may be furnished in the explanation column to make this clear. Example (e) below is such a case.

(a) [A, B, THROUGH Z]

Here the increment value is (1) and the third value in the range is C.

(b) [A, C, THROUGH U]

Here the increment value is (2) and the third value in the range is C.

(c) [AA, BB THROUGH ZZ]

Here the increment values are (1,1) and the third value in the range is CC, that is , each value consists of two identical characters.

(d) [AA, CC, THROUGHYY]

Here the increment values are (2,2) and third value is EE,

(e) [AA, AB, THROUGH ZZ]

Here the increment values are (0,1) and third value is AC. When reaching AZ, the next value will be BA, as the first field (a1) is incremented with 1 (one) when the second field (a2) reaches its maximum. A backup description could be, for example: "Values cycle from AA through AZ, then BA through BZ, etc."

(a) [AA, BA THROUGH ZZ]

Here the increment values are (1, 0) and the third value in the range is CA. When reaching ZA, the next value will be AB, as the second field (a2) is incremented with 1 (one) when the first field (a1) reaches its maximum.

(b) [AA, AB THROUGH ZC]

Here the increment values are (0, 1). The third value in the range is AC and the fourth is BA.

(c) [AA, AC THROUGH ZY]

Here the increment values are (0, 2) and the third value is AE. When reaching AY, the next value will be BA, as the first field (a1) is incremented with 1 (one) when the second field (a2) reaches its maximum. A backup description could be, for example: "Values cycle from AA through AY with increment 2, then BA through BY with increment 2, etc."

(d) [00, 01 THROUGH FF]

Here the increment values are (0, 1). The values cycle from 00 through 0F, then 10 through 1F, then F0 through FF. That is, in this example, the values are hexadecimal.

(e) [AAA, AAB THROUGH ZZZ]

Here the increment values are (0, 0, 1) and the third value is AAC. When reaching AAZ, the next value will be ABA, and when reaching AZZ the next value will be BAA, as the fields will be incremented with 1 (one) when its neighbour position reaches its maximum. This range will have 26 to the third power different values (i.e. 17.576). A backup description could be, for example: "Values cycle from AAA through AAZ, then ABA through ABZ, when reaching AZZ the next cycle will be from BAA through BAZ, then BBA through BBZ, .. when reaching BZZ the next cycle will be from CAA through CAZ, then CBA through CBZ, etc."

(f) [AAA, ABB THROUGH ZZZ]

Here the increment values are (0, 1, 1) and the third value is ACC. When reaching AZZ the next value will be BAA.

(g) [AAAA, AABB THROUGH ZZZZ]

Here the increment values are (0, 0, 1, 1) and the third value is AACC. When reaching AAZZ, the next value will be ABAA, and when reaching AZZZ the next value will be BAAA, as the fields will be incremented with 1 (one) when its neighbour position reaches its maximum. A backup description could be, for example: "Values cycle from AAAA through AAZZ, then ABAA through ABZZ, .. when reaching AZZZ the next cycle will be from BAAA through BAZZ, then BBAA through BBZZ, .. when reaching BZZZ the next cycle will be from CAAA through CAZZ, then CBAA through CBZZ, etc."

(h) [AAAAA, AABAA THROUGH AAZZA]

The first two character positions are constant, which makes them a prefix. The last character position is constant, which makes it a suffix. Therefore only character positions 3 and 4 are handled by these rules, and the increment values for these two character positions are (1, 0). A backup description could be, for example: "Values cycle from AAAAA through AA-ZAA, then AAABA through AAZBA, ... then AAAZA through AAZZA.

(4) Indication of Prohibited Characters. If a range of values is specified and one or more characters which normally would appear in the range are prohibited, this is indicated in the range entry by following the bracketed statement with a parenthetical statement containing OMIT followed by the prohibited characters.

(a) [0000 THROUGH 7777]
(OMIT 8, 9)

Neither 8 nor 9 is permitted in any character position. That is, the values are octal.

(b) [A, B, THROUGH Z]
(OMIT I, O)

Neither I nor O is permitted.

(c) [AA, AB, THROUGH ZZ]
(OMIT I, O)

Neither I nor O is permitted in either character position.

c. Instructive Entry

An instructive entry is recognized by the fact that the first two characters are opening left square brackets ([[). The entry is terminated with two closing right square brackets (]]). Typically, an instructive entry gives a reference and explains or characterizes the nature of data items or data codes. The only instructive entries that are computer processible have the word LITERAL which means, any appropriate value (as indicated in paragraph 401.c) is legal (see examples c.(1) and c.(3) below). Other instructive entries are for the human reader only. Examples of instructive entries are given below.

(1) [[LITERAL]]

This indicates that the data items or data codes are literals.

(2) [[SEE EXPLANATION]]

This would be used if the Explanation column cites a reference document or gives other information for ascertaining data items or codes.

(3) [[LITERAL. SEE EXPLANATION]]

(4) [[SEE REMARKS]]

This page is intentionally left blank

APPENDIX 1
to
ANNEX B

DATA ITEM AND DATA CODE ENTRY SYNTAX

The BNF variant uses the following primitives:

- Non-terminal symbols are enclosed in <>
- Terminal symbols are enclosed in quotes and are capitalized
- Production alternatives are separated by a vertical bar |
- Non-terminal symbols are separated from their productions by ::=
- Optional items are enclosed in square brackets []
- Repeatable items are enclosed in curly brackets {}
- Comments are enclosed by asterisks *

```
<data entry> ::=
          <individual entry>|<range entry>|<instructive entry>

<individual entry>::=
          <individual data item><individual data code>

<range entry>::=

               <range data item>[<omitted characters>]<range data code>[<
          omitted characters>]

               * The number of elements in a data item range must equal the
               number of elements in the corresponding data code range *

<instructive entry>::=
          <instructive data item><instructive data code>

<individual data item> ::=
          {<any character>}

<individual data code>::=
          {<alphanumeric character>}
```

<range data item>::=

```
<numeric range item>[<decimal point locator>]|<alph
anumeric range entry>
```

<range data code> ::=
<numeric range code>
[<decimal point locator>]|<alphanumeric range entry>

<numeric range item> ::=
"["[<item prefix>]<number>[item suffix>]" THROUGH "[<item prefix>]<number>[ite
m suffix]"]"

<numeric range code> ::=
"["[<code prefix>]<number>[code suffix>]" THROUGH "[<code prefix>]<number>[c
ode suffix]"]"

```
<item prefix> ::=
<code prefix>[<blank character>]

<item suffix> ::=
[<blank character>]<code suffix>

<code prefix> ::=
{<alphabetic character>}

<code suffix> ::=
{<alphabetic character>}
```

<decimal point locator> ::=
"("<count>" DECIMAL PLACES)"

<alphanumeric range entry> ::=

"["{<alphabetic and numeric character>}", "{<alphabetic and n
umeric character>}" THROUGH "{<alphabetic and numeric character
>}"]"

<count> ::=

<non zero numeric character>[{<numeric character>}]|{<numeric character>}" TO "{<numeric character>}|{<numeric character>}" OR "{<numeric character>}

<omitted characters> ::=

"(OMIT "<alphabetic and numeric character>[{", "<alphabetic and numeric character>}]")"

<instructive data item>::=
"[["{<any character>}"]]"

** except successive right brackets (]]), and right bracket (]) as last character **

<instructive data code>::=
"[["{<alphanumeric character>}"]]"

<number> ::=
["-"][[{<numeric character>}]"."]{<numeric character>}

<any character> ::=

"/"|":"|"["|"]"|<lower case alphabetic character>|<alphanumeric character>

<alphanumeric character> ::=

<alphabetic and numeric character>|<special character>|<blank character>

<alphabetic and numeric character> ::=
<alphabetic character>|<numeric character>

<alphabetic character> ::=
"A"|"B"|"C" .... "Y"|"Z"

<lower case alphabetic character> ::=
"a"|"b"|"c" .... "y"|"z"

<numeric character> ::=
"0"|<non zero numeric character>

<non zero numeric character>::=
"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

<special character> ::=

```
                    "."|","|"-"|"("|")"|"?"
<blank character>::=
                    " "
```

This page is intentionally blank.

ANNEX C
STRUCTURAL RELATIONSHIPS

1. **Structural Relationships**

Certain restrictions can be placed upon the use of any of the formats that have been designed into a given MTF. These restrictions are the result of internal structural relationships within the message when it is produced by the message drafter. For example, the use of two sets that have been included in the design of an MTF may be restricted to the use of one or the other. That is, they are said to be mutually exclusive. In this example, one of the sets has an occurrence category of operationally determined (O), while the other set has an occurrence category of conditional (C). The condition associated with the latter set is the use or non-use of the former set. All of the restrictions on the use of the various formats for each MTF are documented in Parts II and III in a rigorously defined language. Consequently automated systems can use this language to enforce the restrictions during the message preparation and to validate the structure of a received message to protect against contamination of an operational data base.

The languages in which these restrictions are expressed are defined in the Appendices to this Annex. Appendix 1 defines a language called Structured Language (SL) in which the restrictions are stated in a form of English. Appendix 2 defines a mathematical notation called Structural Notation (SN) in which the restrictions are stated in brief, mathematical expressions. The MTF designer may use either language form to specify the restrictions. ACDBS software is available to translate either form into the other so that both forms will appear in the documentation.

While SN and SL are completely reflexive in the sense that each can be unambiguously translated into the other, it is not always possible to convert the mathematical symbology of SN into a theoretically exact SL text counterpart. For example, what can best be expressed mathematically in SN by an operator (in form of a verb) is sometimes expressed in the text of SL more naturally as part of an operand (in form of a noun). This variation of form is nothing more than that and does not constitute a difference of functionality between SN and SL.

APPENDIX 1 to ANNEX C

STRUCTURED LANGUAGE

1. Purpose

The purpose of this appendix is to specify a computer processable Structured Language (SL) by which the definition of the structure of each message can be completed by consistent application of the rules governing the form and content of messages as constructed of segments, sets, and fields. By utilizing the expressions in this language, automated systems, designers, and managers will have a consistent interpretation of the message design, thereby ensuring that message is structurally correct prior to its transmission. In addition, the automated systems can validate the structure of received messages and thereby avoid possible contamination of their application data bases.

2. Application

2.a. Limitation

Structured Language can only be used to restrict the structure of an MTF to a subset of the legal configurations that are specified by the application of the rules of MTF construction. For example, a format that has an occurrence category of mandatory or operationally determined cannot have its use prohibited by means of the structured language. Similarly, the language can require that a field has a certain value. But that value must be a legal value as defined by the data items and data codes associated with the FUD assigned to that field. Any attempt to use the SL to require a message to be constructed in such a way that it would result in an illegal message is to be considered an illegal MTF design.

2.b. Sequence

The language is applied to complete messages. Therefore, there is no sequence in which the language statements associated with a message must be applied. Nor is there any requirement on the order in which relationships are expressed. That is, restrictions upon formats can be dependent upon features of the message that occur either before or after the format being restricted.

3. Expression

An expression is made up of a statement and an optional condition. The statement specifies the action or requirement, while the condition specifies the conditions under which the action is to occur. Each expression is associated with an MTF. As many expressions as are needed may be associated with a given MTF.

3.a. Statement

A statement defines completely the action that is to occur. It is made up of a left operand with optional subscript, a director, and an optional right operand with optional subscript. The relationships between these five components are depicted in paragraph 5. There are no restrictions on the permitted combinations of operands and directors within a statement except as detailed below.

3.a.(1) Left Operand. The left operand is the recipient of the action. There are five kinds of left operands:

3.a.(1)(a) Set. A set operand is expressed in the following manner:

SET 5 (SETE)

In this example, 5 is the set position. SETE is the ID of that set. It must be enclosed in parentheses. A simple set operand such as this must begin with the word SET.

Note: When the meaning is obvious the single word set may replace the term set position, and field the term field position, in this annex as in other sections of Part I.

3.a.(1)(b) Segment. A segment operand is expressed in one of two ways, depending on the intended meaning. The two ways are explained below by means of examples. In each case 6 is the set position of an initial set of a segment.

SEGMENT 6

This is the usual way to express a segment operand. If the segment is one that may use alternative initial sets, then each occurrence of such a segment is indicated by the first alternative set, regardless of which set is actually used as the initial set of the segment.

SEGMENT ALTERNATIVE 6

This way of expressing a segment operand is only used with segments that have alternative initial sets and will only indicate those segment occurrences which use the indicated alternative initial set. That is, the operand in the example will only apply to segment occurrences with 6 as the initial set, and not to segment occurrences with other alternatives as the initial set.

3.a.(1)(c) Field Group. A repeatable group of fields is expr-essed in the following manner:

THE FIELD GROUP IN SET 5 (SETE)

In this example, 5 is the set position of the set whose field group is being designated as the left operand. It must be preceded by the words THE FIELD GROUP IN, and it must be followed by the set ID enclosed in parenthesis.

3.a.(1)(d) Field. A field is expressed in the following manner:

FIELD 4 IN SET 5 (SETE)

In the example, 4 is the field position in set 5. It is preceded by the word FIELD and followed by the word IN and a set (as described in paragraph 3.a.(1)(a)). If the field happens to be one of a repeating group of fields it may be necessary to specify which occurrence of the repeating group of fields is being referenced. This is done in the following manner:

FIELD 4 IN THE LAST OCCURRENCE OF THE FIELD GROUP IN SET 5 (SETE)

The words THE LAST OCCURRENCE OF are a subscript as described in paragraph 3.c.

3.a.(1)(e) Number Of Occurrences. The number of occurrences of a repeatable field, group of fields, set, or segment is expressed in the following manner:

THE NUMBER OF OCCURRENCES OF

This is followed by the designation of the field, group of fields, set, or segment as indicated above. For example

THE NUMBER OF OCCURRENCES OF SET 10 (SETJ)

In this example, the number of occurrences of set 10 is designated as a left operand.

3.a.(2) Director. The directors indicate the kind of action that is to occur. There are two classes of directors. One class does not utilize a right operand, and the other class requires a right operand.

3.a.(2)(a) No Right Operand. These directors are used with a left operand only and shall have an associated condition, i.e. the left operand must be a format with an occurrence category of "C". They are not used with the FIELD GROUP or the NUMBER OF OCCURRENCES left operand.

IS REQUIRED - Use of the left operand is required within this occurrence of the containing set or segment. If the left operand is a set or segment on level 0 (i.e. not contained within a segment), its use is required within the message. If the left operand is a field, that field must contain a data code that is associated with the field. The use of a hyphen (-) in a required field is illegal. For example:

FIELD 4 IN SET 7 (SETG) IS REQUIRED IF SET 3 OCCURS

In this example, the use of the indicated field is made required whenever set 3 is used.

IS PROHIBITED - Use of the left operand is prohibited within this occurrence of the containing set or segment. If the left operand is a set or segment on level 0 (i.e not contained within a segment), its use is prohibited within the message. If the left operand is a field, that field must not contain a data code. It must contain a hyphen (-) or be omitted by truncation as described in paragraph 509. For example:

SEGMENT 3 IS PROHIBITED IF SET 2 OCCURS

In this example the use of the segment whose initial set is in set position 3 is prohibited whenever set 2 is used.

IS REQUIRED IF ... ,OTHERWISE IT IS PROHIBITED - The associated condition must appear between the words IF and ,OTHERWISE.

SEGMENT 3 IS REQUIRED IF SET 8 (SETH) OCCURS, OTHERWISE IT IS PROHIBITED

In this example the meaning is that the indicated segment must be used if set 8 is used. It is prohibited if set 8 is not used.

3.a.(2)(b) Right Operand Required. These directors require that a right operand be used in the statement. They are only used with the FIELD left operand or THE NUMBER OF OCCURRENCES OF left operand.

IS ASSIGNED THE VALUE - The left operand is assigned the value of the right operand. This director is not used with THE NUMBER OF OCCURRENCES OF left operand. For example

FIELD 4 IN SET 5 (SETE) IS ASSIGNED THE VALUE "ABC"

In this example, ABC is a literal value and is the right operand. Right operands are described in paragraph 3.a.(3). The left operand is FIELD 4 IN SET 5 (SETE). The left operand must be a field, and the right operand must be a literal value. No condition may be associated with this director. A field with an assigned value may not be used as an operand in any condition. A field with an assigned value may be used by automated systems as an aid in the identification of sets in a re-ceived message. For example, several GENTEXT sets may be scheduled in sequence in an MTF. By assigning a value to the first field in each of those sets, a receiving system is able to determine exactly which of the GENTEXT sets have been included in the received message. Another use of this director is to enable automated systems to prepopulate fields in sets, such as the first field in sets MSGID and GENTEXT. In automated systems, an assigned field shall not be overwritten. A received message with a value other than the assigned value in the assigned field shall be treated as an incorrect message.

MUST EQUAL - The left operand must equal or contain the right operand. For example

FIELD 4 IN SET 5 (SETE) MUST EQUAL FIELD 3 IN SET 4 (SETD)

In this example, two fields are required to have identical values other than their field descriptors and blanks used for justification, if any. This also applies to the directors below. The exact meaning is largely depen-dent upon the operands themselves. For instance, if the right operand is a FUD (designated by its FFIRN/FUDN), then that FUD must be selected as the alternative for the field indicated by the left operand.

MUST NOT EQUAL - The left operand must not equal or contain the right operand.

MUST BE GREATER THAN - The left operand must be numerically greater than the right operand.

MUST BE LESS THAN - The left operand must be numerically less than the right operand.

MUST BE SHORTER THAN - The left operand must be shorter (in number of characters) than the right operand. This director may only be used with a variable length field as the left operand and a REMAINING (See paragraph 3.a.(3)(g)) as the right operand.

3.a.(3) Right Operand. The right operand completes the statement. It usually represents a value being imposed upon or compared to the left operand.

3.a.(3)(a) Number of Occurrences. The number of occurrences of a repeatable group of fields, set or segment is expressed in the following manner:

THE NUMBER OF OCCURRENCES OF

These words are followed by the designation of the group of fields, set or segment desired. These are expressed exactly as they are for left operands. The following examples illustrate the number of occurrences of a group of fields, a set, and a segment respectively:

THE NUMBER OF OCCURRENCES OF THE FIELD GROUP IN SET 5 (SETE)

THE NUMBER OF OCCURRENCES OF SET 7 (SETG)

THE NUMBER OF OCCURRENCES OF SEGMENT 9

Note that the number of occurrences of a field may not be used as a right operand.

3.a.(3)(b) Segment, Set, and Field Group. These are used as right operands only in conjunction with the **THIS OCCURRENCE OF** subscript. When so used, they represent an actual integer occurrence number of the repeatable segment, set, or field group. (See paragraph 3.c.(2) **THIS OCCURRENCE OF** for an explanation of operands of these types.)

3.a.(3)(c) Field. A field is expressed in the following manner:

FIELD 4 IN SET 5 (SETE)

This designation is exactly the same as a left operand. The field position, set position, and set ID are indicated. When a field is used as a right operand, it is the contents of the field that are being referenced. For example:

FIELD 1 IN SET 6 (SETF) MUST EQUAL FIELD 4 IN SET 5 (SETE)

In this example, both the left and right operands are fields, with the **MUST EQUAL** director. This means that the value in field one of set six must be identical to the value in field four of set five.

In the event that the set referenced in the right operand is the same set that is referenced in the left operand, then the set reference may be omitted from the right operand. Consider the following example:

FIELD 3 IN SET 6 (SETF) MUST BE GREATER THAN FIELD 2 IN SET 6 (SETF)

Note that both the left and the right operand reference set 6. In this case the right operand can be abbreviated by omitting the set reference. The statement then becomes as follows:

FIELD 3 IN SET 6 (SETF) MUST BE GREATER THAN FIELD 2

3.a.(3)(d) Literal. A literal value is expressed by enclosing the value in quotes. This is used only when the left operand is a field. For example:

FIELD 4 IN SET 5 (SETE) MUST EQUAL "ABCD"

In this example, the indicated field must contain the indicated value. In addition, an asterisk (*) may be used in conjunction with a literal as a wild card. It indicates any string of literal characters. Multiple wild cards may be used and they may precede, be imbedded in, and/or follow the literal value. Note that the asterisk, when used as a wild card, must not be included within the quotes embracing the literal value. For example:

"ABC"*"RST"

This indicates a literal string of six or more characters that begins with **ABC** and ends with **RST**.

3.a.(3)(e) Numeric. A numeric value is expressed simply by the use of numeric characters and, when appropriate, a decimal point. It is used to express number of occurrences, or when referring to a value in a field to which has been assigned a FUD with a numeric range code. For example:

FIELD 3 IN SET 6 (SETF) MUST BE GREATER THAN 45

This example requires that the value in the indicated field exceeds 45. It should be noted that the data code for the FUD assigned to the indicated field must be a numeric range.

3.a.(3)(f) Alternative Contents. This operand is indicated in the following manner:

FFIRN/FUDN 6075-001

It is used to control the selection of alternative contents in a field. For example:

FIELD 3 IN SET 6 (SETF) MUST EQUAL FFIRN/FUDN 6075-001

In this example, the FUD indicated by FFIRN/FUDN 6075-001 must be the alternative contents chosen for use in the field indicated by the left operand.

This operand is only used with the MUST EQUAL and MUST NOT EQUAL directors. Leading zeros for FFIRNs and FUDNs may be omitted.

3.a.(3)(g) Remaining. This operand is indicated in the following manner:

THE NUMBER OF CHARACTERS REMAINING ON THE LINE

This operand is only used with the **MUST BE SHORTER THAN** director to control the size of a variable length field. Usually, that will be one to which has been assigned a FUD whose data code is "literal". For example:

FIELD 4 IN SET 6 (SETF) MUST BE SHORTER THAN THE NUMBER OF CHARACTERS REMAINING ON THE LINE

In this example, the indicated field, field 4, is restricted in length to one character less than the space remaining on the line after the preceding three field positions have been populated.

3.a.(3)(h) Remaining Modified. This operand is an extension of the REMAINING operand described in paragraph 3.a.(3)(g) above. It is indicated in either of the following two ways:

THE NUMBER OF CHARACTERS REMAINING ON THE LINE PLUS 5

THE NUMBER OF CHARACTERS REMAINING ON THE LINE MINUS 5

Any integer may be used with this operand to create an increment or decrement of any size to the number of remaining characters by which the length of the indicated field is to be restricted.

3.b. Condition

A condition specifies the circumstances under which the associated statement is to take effect. It is made up of a left operand with optional subscript, an operator, and an optional right operand with optional subscript. The relationship between these five components are depicted in paragraph 5. An example of an expression with condition is:

SET 3 (SETC) IS REQUIRED IF FIELD 2 IN SET 4 (SETD) EQUALS FIELD 3 IN SET 4 (SETD)

In this example, everything up to the word IF is the statement. The condition, which is always introduced by the word IF, is everything else. Notice that in this example, the use of set 3 is not restricted to be used only when field 2 of set 4 equals field 3 of set 4. It merely states that when field 2 is equal field 3, set 3 must also be used. If it is intended that set 3 be used only when field 2 of set 4 equal field 3 of set 4, then the IS REQUIRED IF ... ,OTHERWISE IT IS PROHIBITED director would be used as shown below:

SET 3 (SETC) IS REQUIRED IF FIELD 2 IN SET 4 (SETD) EQUALS FIELD 3 IN SET 4 (SETD), OTHERWISE IT IS PROHIBITED

The condition in this example is
IF FIELD 2 IN SET 4 (SETD) EQUALS FIELD 3 IN SET 4 (SETD). The director in this example is the only one which causes the condition to be embedded within the statement rather than after it.

A condition is said to be evaluated, returning either true, false, or null. If it returns true, then the associated statement is in effect and is to be enforced. If it returns false or null, the statement is not enforced. The null return differs from the false in that it indicates that the environment for testing the condition does not exist. This null return is significant only in the case of the IS REQUIRED IF ... ,OTHERWISE IT IS PROHIBITED director. A return of null when that director is used causes the entire statement to be ignored.

The example above is a case in point. If field 2 of set 4 does not equal field 3 of set 4, then the condition is evaluated false and the OTHERWISE IT IS PROHIBITED part of the director is activated. That is, the use of set 3 becomes illegal. Now consider the case where set 4 is omitted from the message. A null is returned from the evaluation of the condition since field 2 and 3 of set 4 simply doesn't exist. The environment for testing the condition does not exist, so the condition cannot be evaluated. In this case set 3 becomes neither required nor prohibited; that is, it is treated as if it had an occurrence category of "O".

There are two kinds of conditions: simple and compound.

3.b.(1) Simple Condition. A simple condition is very much like a statement. It is made up of a left operand, an operator, and an optional right operand.

3.b.(1)(a) Left Operand. The left operand is the recipient of whatever action is indicated by the operator. There are five kinds of left operands.

Set. A set left operand of a condition is indicated the same way as a set left operand of a statement.

Segment. A segment left operand of a condition is indicated the same way as a segment left operand of a statement.

Field Group. A field group left operand of a condition is indicated the same way as a field group left operand of a statement.

Field. A field left operand of a condition is indicated the same way as a field left operand of a statement. In the event that the set is the same as the set used in the left operand of the associated statement, the set designator may be omitted. Consider the following example:

FIELD 4 IN SET 5 (SETE) IS REQUIRED IF FIELD 3 IN SET 5 (SETE) OCCURS

Notice that the left operand of the condition, FIELD 3 IN SET 5 (SETE), references the same set (set 5) as the left operand of the statement. In such a case the left operand of the condition can be abbreviated by omitting the set reference:

FIELD 4 IN SET 5 (SETE) IS REQUIRED IF FIELD 3 OCCURS

Like the field operand of a statement, it may be necessary to specify the field group in which the field occurs. Note, however, that the set designator may not be omitted from the field group designation. The condition left operand in the above example would then be as follows:

FIELD 3 IN THE LAST OCCURRENCE OF THE FIELD GROUP IN SET 5 (SETE)

Number of Occurrences. A number of occurrences left operand of a condition is indicated the same way as a number of occurrences left operand of a statement.

3.b.(1)(b) Operator. The operators of conditions are equivalent to the directors of statements. They indicate the kind of action that is being evaluated.

EQUALS - Does the left operand equal or contain the right operand? For example

IF FIELD 4 IN SET 5 (SETE) EQUALS FIELD 3 IN SET 4 (SETD)

In this example the values of the two fields indicated are compared. If they are identical, the condition is evaluated as true.

If the right operand is a FUD, then the assessment is whether or not that FUD has been selected as the alternative content for the field indicated by the left operand.

DOES NOT EQUAL - Does the left operand not equal or contain the right operand?

IS GREATER THAN - Is the left operand greater than the right operand? If the operands are fields, they must both be assigned to FUDs that have a numeric range for a code.

IS LESS THAN - Is the left operand less than the right operand? If the operands are fields, they must both be assigned to FUDs that have a numeric range for a code.

OCCURS - Does the left operand occur? No right operand is used with this operator. For example:

SET 6 (SETF) IS REQUIRED IF SET 5 (SETE) OCCURS

In this example, the condition IF SET 5 (SETE) OCCURS, is evaluated. That is, the message is examined to see if the set has been used. If it has, then the condition is said to return a "true" and the statement is enforced: set 6 must also be used in the message or it is an illegal message.

DOES NOT OCCUR - Does the left operand not occur? No right operand is used with this operator.

3.b.(1)(c) Right Operand. Basically, the right operands of conditions are the same as the right operands of statements. However, the REMAINING operand may not be used in conditions. Also, like the left operand of conditions, if the right operand of the condition and the left operand of the statement are both fields referencing the same set, the set need not be specified in the right operand of the condition. For example

FIELD 2 IN SET 3 (SETC) MUST EQUAL "-" IF FIELD 1 EQUALS FIELD 3

Notice in this example that the set reference has been omitted from both the left and the right operands of the condition because they both refer to SET 3 (SETC), the same set referenced in the left operand of the statement.

An alternative contents right operand shall only be used with the EQUALS and DOES NOT EQUAL operators.

3.b.(2) Compound Condition. A compound condition is one that is made up of two or more simple conditions. The simple conditions are evaluated and combined by the logical operators described below so that only one evaluation, true or false, is returned from the compound condition. Simple conditions connected by the AND logical operator are combined first. Such combinations are then evaluated. The resulting evaluations are then combined with the connecting OR logical operator. This precedence may be altered by the use of parentheses to combine simple conditions. When this is done, the most deeply nested simple conditions are combined first. A null will be returned when a compound condition is used in conjunction with the IS REQUIRED IF ..., OTHERWISE IT IS PROHIBITED director and the non-existence of domains (see paragraph 4) prevents either a true or false evaluation.

3.b.(2)(a) Logical Operator. The simple conditions making up a compound condition are connected by logical operators. There are only two such operators:

AND - This is the logical AND. A compound condition made up of two simple conditions connected by AND evaluates to true only if both simple conditions evaluate true. Consider the following example:

IF FIELD 2 IN SET 5 (SETE) EQUALS "CAN" AND FIELD 3 IN SET 6 (SETF) EQUALS 7

This compound condition will evaluate true only if both field 2 of set 5 contains the value, CAN, and field 3 of set 6 contains the value 7. If either field contains a different value, the compound condition will evaluate false. It should be noted that the compound condition also evaluates to false if either of the simple conditions evaluates to null. The compound condition itself evaluates to null only if all of its simple conditions evaluate to null.

OR - This is the inclusive OR. A compound condition made up of two simple conditions connected by OR is evaluated true if at least one of them is true. Care must be taken with this logical operator when using the negative operators in the simple conditions. The results can be surprising. Consider the following example:

IF FIELD 2 IN SET 6 (SETF) DOES NOT EQUAL "CAN" OR FIELD 2 IN SET 6 (SETF) DOES NOT EQUAL "ADD"

In this example, the compound condition will always evaluate to true. This is because at least one of the two simple conditions must be true regardless of what the value of field 2 might be. A condition such as this should never be used since it accomplishes nothing.

Like the AND logical operator, the compound condition evaluates to null only if all of the simple conditions evaluate to null.

3.b.(2)(b) Multiple Right Operands. A shorthand notation is available when the simple conditions of a compound condition all use the same left operand and the same operator. This is the concept of multiple right operands connected by logical operators. The logical operators to be used are the same as those used to connect the simple conditions. Consider the following example:

IF FIELD 2 IN SET 6 (SETF) DOES NOT EQUAL "CAN" AND FIELD 2 IN SET 6 (SETF) DOES NOT EQUAL "ADD"

Notice that both of the simple conditions use the same operator DOES NOT EQUAL, and both use the same left operand FIELD 2 IN SET 6 (SETF). In this case the repetition of the left operand and the operator can be avoided by using the multiple right operand shorthand notation. The compound condition in the example above using this shorthand notation would then be as follows:

IF FIELD 2 IN SET 6 (SETF) DOES NOT EQUAL "CAN" AND "ADD"

Care must also be taken here because of certain peculiarities in the English language. The shorthand form would seem that it should always return true since field 2 could never have a value of CAN and ADD at the same time. However the expanded form makes the meaning clear. It is suggested that unless the compound condition is so long that it is laborious, the shorthand notation should be avoided when the operator is DOES NOT EQUAL. That is, the shorthand form is recommended for use only with the EQUALS operator.

3.c. Subscript

Any segment may be repeated in a message, as may those sets, field groups, and fields that have been designated as repeatable. (It should be understood that the contents of these segments, sets, field groups, and fields will naturally be different in each repeated occurrence.) When such a segment, set, field group, or field is used as an operand in either a statement or a condition, it is sometimes necessary to specify which occurrence of the segment, set, field group, or field is being referenced. This is done by the use of subscripts. When subscripts are not used, the rules of domain are applied. These rules are discussed in detail in paragraph 4. Generally speaking, the rules of domain mean that corresponding occurrences are paired for comparison or assignment. For example, consider a simple condition comparing the values in two fields of the same set. By default, the comparison is made only between the fields of each occurrence of a possible repeatable set. In addition, if the fields are designated as part of a repeatable group of fields, then the comparison is made for each repetition of the group of fields. This is always the case with columnar sets, because in those sets the fields are always repeatable as a group. When the rules of domain are insufficient and it is necessary to use subscripts, the designators (field, field group, set, and segment) may be used in what is called a path name with a subscript associated with up to three of the designators.

3.c.(1) Subscript Format. Described below are the syntactical formats to be used for the different operand types when it is necessary to use subscripts. In all of the format examples, the following conventions have been used:

Underlines, ___, indicate where a subscript is to be inserted. It always precedes the designator whose occurrence is being specified.

Square brackets, [], enclose omissible elements in the format.

Required portions of the format are in bold face.

For all formats, FIELD 2, SET 5, and SEGMENT 3 designators have been used.

All formats are followed by an example in which all possible subscripts have been included. For illustrative purposes the same subscript, OCCURRENCE <n> OF, has been used and is underlined.

3.c.(1)(a) Field Operand Format. There are three formats for specifying a field operand that requires the use of subscripts on one or more of the four possible designators that may be needed.

- Only Segment Subscripted.

FIELD 2 IN SET 5 (SETE) IN __ SEGMENT 3

Example:

FIELD 2 IN SET 5 (SETE) IN OCCURRENCE 10 OF SEGMENT 3

- Field Group Not Designated.

[___]**FIELD 2 IN**[___]**SET 5 (SETE)**[IN___SEGMENT 3]

Example:

OCCURRENCE 10 OF FIELD 2 IN OCCURRENCE 9 OF SET 5 (SETE) IN OCCURRENCE 8 OF SEGMENT 3

Note that if one of the first two omissible elements is not used, the operand reverts to the format for only the segment subscripted or no subscripts at all.

- Field Group Designated.

FIELD 2 IN ___ THE FIELD GROUP IN[___]**SET 5 (SETE)**[IN___SEGMENT 3]

Example:

FIELD 2 IN OCCURRENCE 10 OF THE FIELD GROUP IN OCCURRENCE 9 OF SET 5 (SETE) IN OCCURRENCE 8 OF SEGMENT 3

Note that whenever a field group designator is used in a field operand, the field group designator is always subscripted, and the field designator is not subscripted.

3.c.(1)(b) Field Group Operand Format. There are only three possible designators that may be modified by subscripts as shown below.

[___]**THE FIELD GROUP IN**[___]**SET 5 (SETE)**[IN___SEGMENT 3]

Example:

OCCURRENCE 10 OF THE FIELD GROUP IN OCCURRENCE 9 OF SETE 5 (SETE) IN OCCURRENCE 8 OF SEGMENT 3

Note that if all of the omissible portions are indeed omitted, what is left is a simple field group operand.

3.c.(1)(c) Set Operand Format. There are only two possible designators that may be modified by subscripts as shown below.

[___]**SET 5 (SETE)**[IN___SEGMENT 3]

Example:

OCCURRENCE 10 OF SET 5 (SETE) IN OCCURRENCE 9 OF SEGMENT 3

Note that if both of the omissible portions are indeed omitted, what is left is a simple set operand.

3.c.(1)(d) Segment Operand Format. There is only one possible designator that may be modified by a subscript as shown below.

[___]**SEGMENT 3**

Example:

OCCURRENCE 10 OF SEGMENT 3

Note that if the omissible portion is indeed omitted, what is left is a simple segment operand.

3.c.(2) Subscript Type. Described below are the subscripts by which the desired occurrences of repeatable designators within operands may be specified.

OCCURRENCE <n> OF - <n> is to be replaced by the desired integer. It indicates a specific occurrence. For example

OCCURRENCE 1 OF FIELD 2 IN SET 6 (SETF)

This operand clearly refers to the first occurrence of field 2 of set 6. Notice that the subscript, OCCURRENCE 1 OF, immediately precedes the designator of the field. Consider the following example:

FIELD 2 IN OCCURRENCE 1 OF SET 6 (SETF)

In this example, the same field operand has been used, but the subscript has been moved to precede the set designator instead of the field designator. The meaning has been completely changed. The reference now is no longer to the first occurrence of the field within all occurrences of the set, but to all occurrences of the field within the first occurrence of the set.

NO OCCURRENCE OF - This subscript can best be described by an example. Given a columnar set with, say, three lines of data, consider the following condition:

IF FIELD 2 IN SET 7 (SETG) EQUALS NO OCCURRENCE OF FIELD 5

This means that for each row in the columnar set, the value in field 2 of the first row will be compared against the value in field 5 of all three rows. Then the value in field 2 of the second row will be compared against the value of field 5 in all three rows. Finally the value in field 2 of the third row will be compared against the value of field 5 in all three rows. Only if none of the comparisons reveal an identical match will the condition return a value of true.

THE LAST OCCURRENCE OF - This subscript references only the last occurrence of an operand. Given the same columnar set used in the previous example with three rows of data, consider the following condition:

IF THE LAST OCCURRENCE OF FIELD 6 IN SET 7 (SETG) EQUALS "ABC"

This condition deals only with the last occurrence of the field. That is, if field 6 in row three contains the value ABC the condition will evaluate to true.

Care must be taken here to note the difference between a non-occurrence of a set or segment and the non-occurrence of a field. In the former case, the set or segment is simply omitted and it is said not to occur. But this is usually not the case with a field. In the example above of field 6 in a columnar set, there may be additional fields. For example, if there is not a value for field 6 but there is one for field 7, then a hyphen must be used in field 6. The hyphen, then, would be considered a non-use or non-occurrence of the field. Therefore, in the example of the columnar set with three rows, if field 6 in the last row contained a hyphen, but field 6 in the preceding row contained a value of ABC, then the condition would still evaluate to true. This is because the last actual occurrence of field 6 was in the second row, not the third row where the field contained a hyphen.

Consider now how the condition should be stated if what is really desired is to test the value of field 6 in the last row of the set, not just the last actual occurrence of the field as was done in the previous example. In this case it would be necessary to specify not which occurrence of which field is to be tested, but which field in which occurrence of the repeating group. This is done as follows:

IF FIELD 6 IN THE LAST OCCURRENCE OF THE FIELD GROUP IN SET 7 (SETG) EQUALS "ABC"

It can be seen that this condition will return a false if the field in row 2 has a value of ABC but the field in row 3, the last occurrence of the field group, has a hyphen.

THE PREVIOUS OCCURRENCE OF - As a complete message is checked for legality, it can be thought of as being processed from beginning to end. During that processing, each element of the message is checked as it is encountered for any structured language expressions that might pertain to it. In this manner, a conceptual current location within the message is constantly maintained. The following example illustrates how this location may be used:

SET 10 (SETJ) IS PROHIBITED IF SET 10 (SETJ) IN THE PREVIOUS OCCURRENCE OF SEGMENT 5 DOES NOT OCCUR

In this example set 10 is a set in segment 5. As the message is processed, when set 10 is encountered the previous occurrence of the segment is examined to see if set 10 was used in that occurrence of the segment. If it was not, then its use is not allowed in the occurrence of the segment that is currently being examined. The effect of this expression is to assure that all occurrences of the segment that utilize set 10 precede the occurrences of the segment that do not utilize the set. That is, once a segment occurs that does not contain a set 10, then none of the following occurrences of the segment may contain a set 10.

SOME OCCURRENCE OF - This subscript is used when all occurrences of a designator in an operand are to be considered in looking for a certain feature. The following example illustrates its use:

FIELD 4 IN SET 5 (SETE) MUST EQUAL SOME OCCURRENCE OF FIELD 3 IN SET 4 (SETD)

In the simplest case to which this example might apply, consider that both set 5 and set 4 are non repeatable sets and not in a segment. Also consider that set 5 does not have a re-peatable group of fields but set 4 does. In this instance the value in field 4 of set 5 is required to match the value in at least one of the occurrences of field 3 in set 4.

THIS OCCURRENCE OF - This is used only to modify a repeatable set, segment, field group or field operand. This subscript shall only be used as the part of a right operand. It means the number of this occurrence. The following example illustrates its use:

FIELD 1 IN SET 10 (SETJ) MUST EQUAL THIS OCCURRENCE OF SEGMENT 10

This means that field 1 of set 10 must contain a value equal to this occurrence number of the segment beginning with set 10. That is, in the third occurrence of the segment beginning with set 10, the value of field 1 in set 10 must be 3.

The use of this subscript to modify a field gives a unique meaning to the field as an operand. In all other cases, a field operand refers to the content of the field. With this subscript, it is not the content but the occurrence number of the field that is the operand.

THE CORRESPONDING OCCURRENCE OF - This subscript is used in conjunction with the subscript SOME OCCURRENCE OF when it is used in a condition. It is used to indicate corresponding occurrences of different operands when such correspondence is non-trivial. That is, the rules of domain discussed in detail in paragraph 4 are insufficient to establish the occurrence correspondence needed. Consider the following example in which correspondence can be considered trivial:

SET 8 (SETH) IS REQUIRED IF SET 7 (SETG) OCCURS

It is considered trivial and therefore unnecessary to specify by subscript that if these are contained in the same segment then the domain of action is within each occurrence of the containing segment. That is, for each occurrence of the segment, if set 7 occurs then set 8 must also occur. However, the presence of set 7 in one occurrence of the segment has nothing to do with the use or non-use of set 8 in a different occurrence of the segment. Consider the following example in which occurrence correspondence must be established by means of the SOME OCCURRENCE OF ... THE CORRE-

SPONDING OCCURRENCE OF subscripts:

FIELD 2 IN SET 6 (SETF) IS REQUIRED IF FIELD 1 EQUALS FIELD 1 IN SOME OCCURRENCE OF THE FIELD GROUP IN SET 5 (SETE) AND FIELD 2 IN THE CORRESPONDING OCCURRENCE OF THE FIELD GROUP IN SET 5 (SETE) DOES NOT OCCUR

In this example there is a compound condition. In the first simple condition, all occurrences of the field group in set 5 are searched for a value in field 1 that matches that of field 1 in set 6. When a match is found, field 2 in that same occurrence (the "corresponding" occurrence) of the field group is examined. If it does not occur, then the compound condition is evaluated as true and field 2 in set 6 is considered required.

When this operand is used in a statement, nothing is needed for the left operand while THE CORRESPONDING OCCURRENCE OF appears as a subscript in the right operand. See Rule 2 in paragraph 4 for an example of the use of this operand in a statement.

4. Domain

The domain is the range over which the operands of an expression are applied. The simplest and most intuitively obvious of such domains is the message itself. For example:

SET 6 (SETF) IS REQUIRED IF SET 5 (SETE) OCCURS

The condition refers to the use of set 5 within the same occurrence of the message in which set 6 is being considered for use. The fact that set 5 may have been used in some other message occurrence of the same MTF is irrelevant to the message at hand. This notion of domain is used elsewhere in the standard as, for example, in the occurrence category of fields. In that case the domain is not the entire message occurrence, but only the occurrence of the containing set. That is, an occurrence category of M does not mean that the field is required in every occurrence of the message, but only that it is required in every occurrence of the set. The domain of the field occurrence category is said to be the occurrence of its containing set.

In a similar manner, the domain for a given set is the message containing it unless the set is in a segment. In this latter case the domain is the occurrence of the containing segment.

Usually, all operands in an expression will be in the same domain. When this is not the case, domains can be modified through the use of subscripts as described in paragraph 3.c. Since fields may exist within repeating groups of fields, and repeating groups of fields may exist within repeating sets, and repeating sets may exist within repeating segments, it is necessary to be specific about exactly which occurrences are to be included in the interpretation of any structured language expression. For example:

FIELD 5 IN SET 5 (SETE) MUST EQUAL FIELD 6 IN SET 5 (SETE)

In this example, if set 5 is repeatable the question arises as to which field 6 value must the field 5 value match. Even though it seems intuitively obvious that field 5 should match field 6 <u>within the same occurrence of the set</u>, if some kind of rules of domain were not applied (even subconsciously), it would be necessary to use subscripts to assure that the correct interpretation is made. Such a use of subscripts would become extremely laborious, since many operands would require complete path names (field group, set, and segment) with subscripts on all designators. In the case of nested segments, it would be necessary to provide a subscript for each level of nesting. Rather than be burdened with this excessive subscripting, it is preferable to establish the certain rules of domain. For the most part, these rules are intuitively obvious, like the example above. Nevertheless, they must be specified.

<u>Rule 1.</u> The left operand of a statement is applied to all occurrences of that operand, one at a time, in the message. For example, if the left operand of a statement is **FIELD 2 IN SET 4 (SETD)**, each occurrence of set 4 field 2 will be evaluated according to the rest of the expression. The domain of the left operand is the extent of its containing format occurrence. For example:

SET 4 (SETD) IS REQUIRED

This example means that if set 4 is contained within a segment, it is required each time that segment is used. If the segment is not used, then neither is the set. A statement such as the one in this example shall have a condition associated with it.

<u>Rule 2.</u> The right operand of a statement, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of the statement. For example:

FIELD 1 IN SET 8 (SETH) MUST EQUAL FIELD 1 IN SET 7 (SETG)

If both sets are in the same segment, then within each occurrence of the segment field 1 of set 8 must match field 1 of set 7. Note that if set 8 is repeatable, then all occurrences of set 8 must have a field 1 that matches field 1 of set 7 within that occurrence of the containing segment. But if set 7 is repeatable within the segment, the statement becomes meaningless because it is not known which occurrence of set 7 within the segment occurrence is being referenced. If it is intended that field 1 in the first occurrence of set 7 is to be matched by field 1 in the first occurrence of set 8 and so forth through all occurrences of both sets, then the subscripts must be used in this manner:

FIELD 1 IN SET 8 (SETH) MUST EQUAL THE CORRESPONDING OCCURRENCE OF FIELD 1 IN SET 7 (SETG)

Notice that this is a use of the subscript THE CORRESPONDING OCCURRENCE OF which is not in conjunction with the use of the SOME OCCURRENCE OF subscript. This is because the occurrence to which an occurrence of a different operand must correspond is the left operand of the statement. By Rule 1 above, all occurrences of the statement left operand must be considered, so a "corresponding" indication would be meaningless for the statement left operand. However, correspondence between occurrences of sets 7 and 8 must be established by use of subscripts. Another way in which the correspondence could have been accomplished by the MTF designer would be to group sets 7 and 8 into a repeating nested segment rather than letting each set be repeatable independently within a segment. If such a nested segment had been created, then, by Rule 2, the use of THE CORRESPONDING OCCURRENCE OF subscript would be unnecessary.

Rule 3. The left operand of a condition, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of the statement. For example:

FIELD 4 IN SET 8 (SETH) IS REQUIRED IF FIELD 2 OCCURS

If field 2 and field 4 are in a repeating group of fields, in each occurrence of the group there must be a value for field 4 if there is a value for field 2 in that same occurrence of the group of fields. Now consider the same example in the case where field 4 is contained in the repeating group of fields but field 2 is not. In this case, a value in field 2 will require that a value occurs in field 4 for all occurrences of the repeating group of fields. This is because field 2, the left operand of the condition, is in the direct line of nesting (one level up) to all occurrences of the repeating group of fields that contains field 4, the left operand of the statement. Therefore, that one occurrence of field 2 renders field 4 required in all possible occurrences of field 4 (all occurrences of the repeating group of fields.)

Rule 4. The right operand of a condition, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of the condition. For example:

IF FIELD 2 IN SET 8 (SETH) EQUALS FIELD 4

In this condition, the comparison of fields takes place only within the same occurrence of set 8. If, by Rule 3, several occurrences of set 8 must be evaluated, they must all evaluate true in order to return true for the condition. Within a given occurrence of set 8, if field 4 is repeatable and field 2 is not, all of the values of field 4 must be equal to that of field 2 in order for the condition to evaluate true. Note, however, that since the repeating group of fields has not been indicated, only those occurrences of the repeating group of fields in which field 4 has a value (not a hyphen) will be considered. If it is intended that field 4 in all occurrences of the repeating group of fields must equal the value in field 2 in order to return a true evaluation, the condition should be written as follows:

IF FIELD 2 IN SET 8 (SETH) EQUALS FIELD 4 IN THE FIELD GROUP IN SET 8 (SETH)

By specifying the repeating group explicitly, all occurrences of the repeating group are qualified for examination. If it is intended that a true should be returned if field 2 matches any occurrence of field 4 then the right operand of the condition should be modified by a subscript as follows:

IF FIELD 2 IN SET 8 (SETH) EQUALS FIELD 4 IN SOME OCCURRENCE OF THE FIELD GROUP IN SET 8 (SETH)

In this case the same result would be obtained by subscripting the field designator rather than the field group designator and omitting the latter as shown in the example below:

IF FIELD 2 IN SET 8 (SETH) EQUALS SOME OCCURRENCE OF FIELD 4

Note that if this form is chosen (field group not specified) the abbreviated form of the field operand can be used in which the set identifier need not be repeated. (See paragraph 3.a.(3)(c).)

5. Component Relationships of Expressions

The following pages contain one table that depicts relationships for the statement portion of expressions and another table that depicts relationships for the condition portion of expressions.

Component relationships for STATEMENT portion of expressions

SUBSCRIPT	LEFT OPERAND	DIRECTOR	SUBSCRIPT	RIGHT OPERAND
	-segment -set -field	-is required **(1** -is prohibited **(1** -is required if ..., otherwise it is pro-hibited **(1**		
	-field	-is assigned the value **(2**		-literal
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-must equal -must be greater than -must be less than -must not equal	-this occurrence of **(3**	-segment -set -field group
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-must equal -must be greater than -must be less than -must not equal	-no occurrence of -the last occurrence of -the previous occur-rence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of -this occurrence of	-field
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-must equal -must be greater than -must be less than -must not equal		-number of occur-rences (used with segment, set and field group)
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-must equal -must be greater than -must be less than -must not equal		-numeric

Component relationships for STATEMENT portion of expressions (continued)

SUBSCRIPT	LEFT OPERAND	DIRECTOR	SUBSCRIPT	RIGHT OPERAND
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding occurrence of	-field	-must equal -must not equal		-literal -alternative contents
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding occurrence of	-field	-must be shorter than		-remaining -remaining modified
	-number of occurrences (used with segment, set, field group and field)	-must equal -must be greater than -must be less than -must not equal	-this occurrence of **(3**	-segment -set -field group
	-number of occurrences (used with segment, set, field group and field)	-must equal -must be greater than -must be less than -must not equal	-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding occurrence of -this occurrence of	-field
	-number of occurrences (used with segment, set, field group and field)	-must equal -must be greater than -must be less than -must not equal		-number of occurrences (used with segment, set and field group)
	-number of occurrences (used with segment, set, field group and field)	-must equal -must be greater than -must be less than -must not equal		-numeric

1) This statement form is only used if a condition is present.
2) This statement form is only used if no condition is present.
3) Subscript is mandatory.

Component relationships for CONDITION portion of expressions

SUBSCRIPT	LEFT OPERAND	OPERATOR	SUBSCRIPT	RIGHT OPERAND
-the corresponding oc-currence of	-segment -set -field group -field	-occurs -does not occur		
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-equals -is greater than -is less than -does not equal	-this occurrence of **(1**	-segment -set -field group
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-equals -is greater than -is less than -does not equal	-no occurrence of -the last occurrence of -the previous occur-rence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of -this occurrence of	-field
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-equals -is greater than -is less than -does not equal		-number of occur-rences (used with segment, set and field group)
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding oc-currence of	-field	-equals -is greater than -is less than -does not equal		-numeric

Component relationships for CONDITION portion of expressions (continued)

SUBSCRIPT	LEFT OPERAND	LOGICAL OPERATOR	SUBSCRIPT	RIGHT OPERAND
-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding occurrence of	-field	-equals -does not equal		-literal -alternative contents
	-number of occurrences (used with segment, set, field group and field)	-equals -is greater than -is less than -does not equal	-this occurrence of **(1**	-segment -set -field group
	-number of occurrences (used with segment, set, field group and field)	-equals -is greater than -is less than -does not equal	-no occurrence of -the last occurrence of -the previous occurrence of -some occurrence of -occurrence <n> of -the corresponding occurrence of -this occurrence of	-field
	-number of occurrences (used with segment, set, field group and field)	-equals -is greater than -is less than -does not equal		-number of occurrences (used with segment, set and field group)
	-number of occurrences (used with segment, set, field group and field)	-equals -is greater than -is less than -does not equal		-numeric

1) Subscript is mandatory.

6. Syntax of Structured Language[2]

The BNF variant used in this appendix uses the following primitives:

- Non-terminal symbols are enclosed in <>.

- Terminal symbols are enclosed in quotes and capitalized.

- Production alternatives are separated by a vertical bar *.

- Non-terminal symbols are separated from their productions by ::=.

- Optional items are enclosed in square brackets [].

- Repeatable items are enclosed in curly brackets {}.

- Comments are enclosed in asterisks ** and are written in italics.

  E.g. **This is a comment**

```
<expression>::=
        <statement>[" IF "<condition>]|<statement simple operand>
        <unary statement director>" IF "<condition>|
        <statement simple operand>" IS REQUIRED IF "<condition>",
        OTHERWISE IT IS PROHIBITED"|
        <statement field id>" IS ASSIGNED THE VALUE "<literal operand>

<statement>::=
        <statement left operand><dyadic statement director>
        <right operand>|
        <statement left operand><alternative dyadic statement director>
        <alternative right operand>|
        "THE NUMBER OF OCCURRENCES OF " <statement simple operand>
        <dyadic statement director><right operand>|
        "THE NUMBER OF OCCURRENCES OF "<field group id>
        <dyadic statement director><right operand>|
        <statement left operand>" MUST BE SHORTER THAN "
        <remainder operand>

<condition>::=
        <simple condition>|"("<condition>")"|
```

[2] The updated paragraph is documented in Annex F to the Case Law Document

```
                <condition><logical operator><condition>

<simple condition>::=
                ["THE CORRESPONDING OCCURRENCE OF "]
                <condition simple operand><unary condition operator>|
                <condition left operand><dyadic condition operator>
                <compound right operand>|
                <condition left operand><alternative dyadic condition operator>
                <extended compound right operand>|
                "THE NUMBER OF OCCURRENCES OF "
                <condition simple operand><dyadic condition operator>
                <right operand>

<condition left operand>::=
                <subscript><subscripted condition field id>|<condition field id>

<statement left operand>::=
                <subscript><subscripted statement field id>|<statement field id>

<compound right operand>::=
                <right operand>[{" AND "<right operand>}]|<right operand>
                [{" OR "<right operand>}]

<extended compound right operand>::=
                <extended right operand>[{" AND "<extended right operand>}]|
                <extended right operand>[{" OR "<extended right operand>}]

<extended right operand>::=
                <right operand>|<alternative right operand>

<right operand>::=
                <subscript><subscripted condition field id>|<condition field id>|
                "THE NUMBER OF OCCURRENCES OF "
                <no field simple operand>|
                "THIS OCCURRENCE OF "<condition simple operand>|<integer>

<alternative right operand>::=
                <literal operand>|<wildcard operand>|<fud operand>

<statement simple operand>::=
                <segment id>|<set id>|<statement field id>

<condition simple operand>::=
                <segment id>|<set id>|<field group id>|<condition field id>

<no field simple operand>::=
                <segment id>|<set id>|<field group id>

<wildcard operand>::=
                [<literal operand>]{"*"<literal operand>}["*"]|<literal operand>"*"
```

```
<fud operand>::=
          "FFIRN/FUDN"<integer>"-"<integer>

<remainder operand>::=
          "THE NUMBER OF CHARACTERS REMAINING ON THE LINE"[<increment><int
          eger>]

<literal operand>::=
          """<literal>"""
               *This produces a literal in quotation marks.*

<statement field id>::=
     "FIELD "<integer>" IN "<subscript><field group id>|<subscripted statement field id>

<condition field id>::=
          "FIELD "<integer>" IN "<subscript><field group id>|
          <subscripted condition field id>

<subscripted statement field id>::=
          "FIELD "<integer>" IN "[<subscript>]<set id>

<subscripted condition field id>::=
          "FIELD "<integer>|"FIELD "<integer>" IN "[<subscript>]<set id>

<field group id>::=
          "THE FIELD GROUP IN "[<subscript>]<set id>

<set id>::=
          "SET "<integer>" ("<set name>")"[" IN "<subscript><segment id>]

<segment id>::=
          "SEGMENT "["ALTERNATIVE "]<integer>
          [{" IN "<subscript><segment id>}]

<set name>::=
          <literal>

<subscript>::=
          "NO OCCURRENCE OF "|"THE LAST OCCURRENCE OF "|
          "THE PREVIOUS OCCURRENCE OF "|
          "SOME OCCURRENCE OF "|"OCCURRENCE "<integer>" OF "|
          "THE CORRESPONDING OCCURRENCE OF "

<increment>::=
          " PLUS "|" MINUS "

<logical operator>::=
          " AND "|" OR "

<unary condition operator>::=
          " OCCURS"|" DOES NOT OCCUR"
```

```
<dyadic condition operator>::=
            " EQUALS "|" IS GREATER THAN "|" IS LESS THAN "|
            " DOES NOT EQUAL "
<alternative dyadic condition operator>::=
            " EQUALS "|" DOES NOT EQUAL "

<unary statement director>::=
            " IS REQUIRED"|" IS PROHIBITED"

<dyadic statement director>::=
            " MUST EQUAL "|" MUST BE GREATER THAN "|
            " MUST BE LESS THAN "|" MUST NOT EQUAL "

<alternative dyadic statement director>::=
            " MUST EQUAL "|" MUST NOT EQUAL "

<literal>::=
            {<char>}

<char>::=
            <digit>|<special char>|" "|"A"|"B"|"C"|....|"X"|"Y"|"Z"

<integer>::=
            {<digit>}

<digit>::=
            "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

<special char>::=
            "."|","|"-"|"("|")"|"?"
```

THIS PAGE IS INTENTIONALLY BLANK

APPENDIX 2
To
ANNEX C

STRUCTURAL NOTATION

1. <u>Purpose</u>

The purpose of this appendix is to specify a computer processable Structural Notation (SN) by which the definition of the structure of each message can be completed by consistent application of the rules governing the form and content of messages as constructed of segments, sets, and fields. By interpreting the notation, automated systems, designers and managers will have a consistent interpretation of the message design thereby ensuring that each MTF is structurally correct prior to its transmission. In addition, automated systems can validate the structure of received messages and thereby avoid possible contamination of their application data bases. A structured language (SL) equivalent can be derived from the notation to provide readability to the user of the documentation.

2. <u>Application</u>

2.a. <u>Limitation</u>

Structural notation can only be used to restrict the structure of an MTF to a subset of the legal configurations that are specified by the application of the rules of MTF construction. For example, a format that has an occurrence category of mandatory or operationally determined cannot have its use prohibited by means of the structural notation. Similarly, the notation can require that a field has a certain value. But that value must be a legal value as defined by the data items and data codes associated with the FUD assigned to that field. Any attempt to use the structural notation to require a message to be constructed in such a way that it would result in an illegal message is to be considered an illegal MTF design.

2.b. <u>Sequence.</u>

The notation is applied to complete messages. Therefore, there is no sequence in which the notation statements associated with a message must be applied. Nor is there any requirement on the order in which the conditionalities are expressed. That is, restrictions upon formats can be dependent upon other features of the message that occur either before or after the format being restricted.

3. Expression

A mathematical expression is made up of a statement and an optional condition. The statement specifies the action or requirement, while the condition specifies the conditions under which the action is to occur. Similar restrictions to the construction of statements and conditions as those which apply to SL (see Appendix 1 paragraph 5) also apply to SN. Each expression is associated with an MTF. As many expressions as are needed may be associated with a given MTF.

3.a. Statement

A statement defines completely the action that is to occur. It is made up of a left operand, a director, and an optional right operand. There are no restrictions on the permitted combinations of operands and directors within a statement except for the specific restrictions on the use of certain left operands detailed in the following operand descriptions.

3.a.(1) Left Operand

The left operand is the recipient of the action. There are four kinds of left operands.

3.a.(1)(a) Set. <(n)> - A set operand is expressed by enclosing the set position in parentheses. The set position is the sequence number of the set as it occurs in the MTF. For example, (5) indicates the fifth set in the MTF as specified in its design, that is the set position five.

Note: When the meaning is obvious the single word set may replace the term set position and field the term field position in this annex as in other sections of Part I.

3.a.(1)(b) Segment. <(nS)> or <(nI)> - A segment operand is expressed by enclosing in parentheses the set position of an initial set of the segment followed by S or I. The S is used to indicate segments that do not have alternative initial sets and to indicate segments that use any of their legal alternative initial sets. The I is used to indicate only those segments that use the particular alternative initial set indicated by the set position. For example, given a segment that has alternative initial sets 6 and 7. In that case (6S) indicates each occurrence of the segment, regardless of whether a given segment occurrence actually use set 6 or 7 as the initial set. On the other hand, (6I) indicates only those occurrences of the segment that actually use set 6, and (7I) only those that use set 7.

3.a.(1)(c) Field Group. <(n)FG> - A repeatable group of fields is expressed by following the set expression with FG. For example, (5)FG indicates the repeatable group of fields in set 5.

3.a.(1)(d) Field. <(n)Fm> - A field is expressed by the containing set expression followed by F and the field position. For example, (5)F4 indicates the fourth field of set 5. If the field happens to be one of repeatable group of fields it may be necessary to specify which occurrence of the repeatable group of fields is being referenced. This is done by expressing the field group as described in paragraph 3.a.(1)(c) above instead of simply specifying the set. For example, (5)FG,LF4 indicates field 4 in the last occurrence of the repeatable group of fields in set 5. The ,L is a subscript. It is described in paragraph 3.c.

3.a.(2) Director

The directors indicate the kind of action that is to occur. There are two classes of directors. One class does not utilize a right operand and the other class requires a right operand.

3.a.(2)(a) No Right Operand. These directors are used with a left operand only and shall have an associated condition, i.e. the left operand must be a format with an occurrence category of "C". They are not used with the FIELD GROUP left operand.

Q Use of the left operand is required within this occurrence of the containing set or segment. If the left operand is a set or segment on level 0 (not contained within a segment), its use is required within the message. If the left operand is a field, that field must contain a data code that is associated with the field. The use of a hyphen (-) in a required field is illegal.

P Use of the left operand is prohibited within this occurrence of the containing set or segment. If the left operand is a set or segment on level 0 (not contained within a segment), its use is prohibited within the message. If the left operand is a field, that field must not contain a data code. It must contain a hyphen (-) or be omitted by truncation as described in paragraph 509.

QP Use of the left operand is required within this occurrence of the containing set or segment if the associated condition evaluates true, otherwise its use is prohibited. The meaning of required and prohibited are as described above for the directors Q and P.

3.a.(2)(b) Right Operand Required. These directors require that a right operand be used in the statement.

A The left operand is assigned the value of the right operand. The left operand must be a field and the right operand must be a literal value. No condition may be associated with a statement using the A director. A field with an assigned value may not be used as an operand in any condition. A field with an assigned value may be used by automated systems as an aid in the identification of sets in a received message. For example, several GENTEXT sets may be scheduled in sequence in an MTF. By assigning a value to the first field in each of those sets, a receiving system is able to determine exactly which of the GENTEXT sets have been included in the received message. Another use of this director is to enable automated systems to pre-

	populate fields in sets, such as the first field in sets MSGID and GENTEXT. In automated systems, this field should not be over - writable. A received message with a value other than the assigned value should be treated as an incorrect message.
=	The left operand must equal or contain the right operand. The exact meaning is largely dependent upon the operands themselves. For example, if the right operand is a FUD (designated by its FFIRN/FUDN), then that FUD must be selected as the alternative content for the field indicated by the left operand.
!=	The left operand must not equal or contain the right operand.
>	The left operand must be numerically greater than the right operand.
<	The left operand must be numerically less than the right operand.
@=	The left operand must occur the number of times indicated by the right operand.
@>	The left operand must occur greater than the number of times indicated by the right operand.
@<	The left operand must occur less than the number of times indicated by the right operand.
@!	The left operand must not occur the number of times indicated by the right operator.
?<	The left operand must be less in length than indicated by the right operand. The principal use of this director is to restrict the size of a field to something less than the FUD would normally allow.

3.a.(3) Right Operand

The right operand completes the statement. It usually represents a value being imposed or compared to the left operand.

3.a.(3)(a) <u>Set</u>. <[n]> - A set operand is expressed by enclosing the set position in square brackets. Unless modified by a subscript (see paragraph 3.c below), it is interpreted to mean the number of occurrences of the set within its containing segment, if any.

3.a.(3)(b) <u>Segment</u>. <[nS]> or <[nI]> - A segment operand is expressed by enclosing in square brackets the set position of the initial set of the segment followed by S or, in case of a segment with alternative initial sets, the set position of one of the alternative initial sets followed by I. The selection of the S or I is dependent upon what is intended with regard to segments with alternative initial sets as described for left operands. (See paragraph 3.a.(1)(b)). Like the set, this usually means the number of occurrences of the segment within its containing segment.

3.a.(3)(c) <u>Field Group</u>. <[n]FG> - A group of fields operand is expressed by following the set designator with FG. Like the set and segment, this usually means the number of occurrences of the repeatable group of fields within its containing set.

3.a.(3)(d) <u>Field</u>. <[n]Fm> - A field is expressed by the containing set designator followed by F and the field position. If the set is the same as the set specified in the left operand, then the set designator may be omitted from the right operand. For example, (6)F3 > F2 means that the value in field 3 of set 6 must be greater than the value of field 2 of the same set. If the field happens to be one of a repeating group of fields it may be necessary to specify which occurrence of the repeating group of fields is being referenced. This is done by designating the field group described in paragraph 3.a.(3)(c) above instead of simply specifying the set. For example, [5]FG,LF4 indicates field 4 in the last occurrence of the repeating group of fields in set 5. The ,L is a subscript. It is described in paragraph 3.c.

3.a.(3)(e) Literal. <"literal"> - A literal value is expressed by enclosing it in quotes. This is used only when the left operand is a field. It represents a potential or actual field literal value. For example, (5)F4="ABCD" means that field 4 in set 5 must contain ABCD. An asterisk (*) may be used in conjunction with a literal as a wild card of zero or more characters. For example, "ABC"* indicates any literal value beginning with ABC. In addition to following the literal value as shown in the example, a wild card may precede and/or be embedded in a literal value. Note that the wild card must not be included within the quotes (" "). For example "ABC"*"RST" indicates any value beginning with ABC and ending with RST.

3.a.(3)(f) Coded Value. <"code"> - The value for a field which is associated with a FUD that has a specified list of codes is indicated exactly the same as a literal.

3.a.(3)(g) Numerics. <n> - Numeric information is expressed simply by the appropriate numeric notation. This may express an actual numeric value, a number of occurrences, or an actual occurrence number as determined by the rest of the statement.

3.a.(3)(h) Alternative Contents. <FFnnnn-nnn> - A FUD is indicated by FF followed by the FFIRN followed by - followed by the FUDN. Leading zeros for both numbers may be omitted. This is used to specify a specific FUD for use in an alternative contents field.

3.a.(3)(i) Remaining. <R> - This single character operand means the number of spaces remaining on the line. It is only used in conjunction with the director ?<.

3.a.(3)(j) Remaining, Modified. <R±n> - This is the same as paragraph 3.a.(3)(i) above but arithmetically modified by a numeric increment or decrement.

3.b. Condition

A condition specifies the circumstances under which the associated statement is to take effect. For example, when a set must always be used when another set is used, one might say "set 6 is required if set 5 is used". In this example, "if set 5 used" is the condition associated with the statement, "set 6 is required". A condition is said to be evaluated, returning either true, false, or null. If it returns true, the statement is in effect and should be enforced. If it returns false or null, the statement is not enforced. The null return differs from the false in that it indicates that the domain of the simple condition does not exist. This is significant only with the QP director. A return of false activates the "prohibited" feature of the director. A return of null should cause the entire statement to be ignored. For example:

(3) QP ([4]F2 = [4]F3)

This states that set 3 is required if field 2 of set 4 equals field 3 of set 4. Consider the case in which fields 2 and 3 are not equal. In that case a false would be returned and the use of set 3 would be prohibited. Now consider the case where set 4 is not used in the message. Fields 2 and 3 of set 4 cannot be compared because set 4 does not exist. In this instance a null would be returned because the domain in which the condition operates does not exist. The entire statement is then to be ignored. That is, set 3 is neither required nor prohibited as far as this statement is concerned. Domains are discussed in more detail in paragraph 4.

In the notation, the entire condition is enclosed in parentheses and immediately follows its associated statement. Note that statements can exist without conditions, but a condition must always be associated with a single statement.

There are two kinds of conditions: simple and compound.

3.b.(1) Simple Condition

A simple condition is, in form, very much like a statement. It is made up of a left operand, an operator, and an optional right operand. It must be enclosed in parentheses.

3.b.(1)(a) Left Operand. The left operand is the recipient of whatever action is indicated by the operator. There are four kinds of left operands.

Set.	<[n]> - A set operand is the same as a statement set operand except that is enclosed in square brackets instead of parentheses. This operand may also be modified by a subscript. (See paragraph 3.c.)
Segment.	<[nS]> or <[nI]> - A segment operand is the same as a statement segment operand except that is enclosed in square brackets instead of parentheses. This operand may also be modified by subscripts. (See paragraph 3.c.)
Field Group.	<[n]FG> - A field group operand is the same as a statement field group operand except that the set position is enclosed in square brackets instead of parentheses. This operand may also be modified by a subscript. (See paragraph 3.c.)
Field.	<[n]Fm> - A field operand is the same as a statement field operand except that the set position is enclosed in square brackets instead of parentheses. In the event that the set is the same as the set used in the left operand of the associated statement, the set designator may be omitted. Like the statement operand, it may be necessary to specify the repeating group occurrence of the field.

3.b.(1)(b) Operator. The operators are equivalent to the directors in statements. They indicate the kind of action that is being assessed.

=	Does the left operand equal or contain the right operand? The exact meaning is largely dependent upon the operands themselves. For example, if the right operand is a FUD, then the assessment is if that FUD has been selected as the alternative content for the field indicated by the left operand.
!=	Does the left operand not equal or contain the right operand?
@=	Does the left operand occur the number of times indicated by the right operand?
@>	Does the left operand occur a greater number of times than indicated by the right operand?

@<	Does the left operand occur a lesser number of times than indicated by the right operand?
@!	Does the left operand not occur the number of times indicated by the right operand?
>	Is the left operand greater than the right operand?
<	Is the left operand less than the right operand?
@	Does the left operand occur? No right operand is used.
!@	Does the left operand not occur? No right operand is used.

3.b.(1)(c) Right Operand. The right operands are the same as the right operands of statements. Note that when a field is used the set designator may be omitted if it is the same set as the left operand of the statement. For example, (3)F2 = "-" (F1 = F3). This means that field 2 of set 3 must be a hyphen (-) if field 1 of that set is equal to field 3 of that set.

3.b.(2) Compound Condition

A compound condition is one that is made up of two or more simple conditions. The simple conditions are evaluated and combined by the logical operators described below so that only one evaluation, true or false, is returned from the compound condition. Simple conditions connected by the & are combined first. This precedence may be altered by the use of parentheses to combine simple conditions. When this is done the most deeply nested simple conditions are combined first. A null will be returned when a compound condition is used in conjunction with the QP director and the non-existence of domains (see paragraph 4) prevents either a true or false evaluation.

3.b.(2)(a) Logical Operator. The simple conditions making up a compound condition are connected by logical operators. There are only two such logical operators.

& This is the logical and. A compound condition made up of two simple conditions connected by & is evaluated true only if both simple conditions evaluate true. For example, (((5)F2 = "CAN") & ((6)F3 = 7)) will evaluate true only if field 2 of set 5 contains CAN and field 3 of set 6 contains 7.

/ This is the inclusive or. A compound condition made up of two simple conditions connected by / is evaluated true if at least one of them is true. Care must be taken in using the logical operator when operators of the simple conditions are a "not" form, for the results can be surprising. For example, (((6)F2 != "CAN") / ((6)F2 != "ADD")) will always evaluate true. This is because at least one of the simple conditions must be true, regardless of what the value of F2 actually is.

3.b.(2)(b) Multiple Operands. A shorthand notation is available when the simple conditions of a compound condition all use the same operator. This is the concept of multiple right operands connected by logical operators. The logical operator to be used is the same as that which would be used to connect the simple conditions. For example, (((6)F2 != "CAN") & ((6)F2 != "ADD")) can be abbreviated as follows: ((6)F2 != "CAN" & "ADD"). This can also be confusing. At first it appears that the multiple operand form in the example should always return true because the same field cannot be both ADD and CAN at the same time. The expanded form, however, makes it clear that if the field has either of the indicated values, one of the simple conditions will be false, thus evaluating the entire compound condition to false.

3.c. Subscript

Any segment may be repeated in a message, as may those sets, field groups, and fields that have been designated as repeatable. When such a segment, set, field group, or field is used as an operand (in either a statement or condition), it is sometimes necessary to specify which occurrence of the segment, set, field group, or field is being referenced. This is done by means of subscripts. The subscript follows the segment, set, field group or field designator and is separated from it by a comma (,). When subscripts are not used, the rules of domain are used. These are discussed in detail in paragraph 4. Generally speaking, this means that corresponding occurrences are paired for comparison or assignment. For example, (F2 = F3) is a simple condition comparing field 2 with field 3 within the same set. By default this comparison is made only between the fields of each occurrence of a possibly repeatable set. In addition, if the fields are designated part of a repeatable group of fields, then the comparison is made for each repetition of the group of fields. This is always the case with columnar sets, because in those sets the fields are always repeatable as a group.

<n> The n^{th} occurrence. For example, [6]F2,1 indicates the first occurrence of the second field of set 6. Note that the reference is to the field, not to the set. [6],1F2 indicates field 2 of the first occurrence of set 6. Care should be taken to note the difference between the occurrence of the field and the field group. In the example shown, if the repeatable group of fields occurs 5 times, but field 2 contains a value (not a hyphen) in only the third and fourth occurrences of the group of fields, then

the reference is to field 2 in the third occurrence of the field group. If it is desired to reference field 2 in the first occurrence of the repeatable group of fields, it should be done in this way: [6]FG,1F2. It is only rarely necessary to draw this distinction.

0 The zeroth occurrence. By convention, this is interpreted to mean "no occurrence". For example, (F2,0 = "CAN") is the condition in which no occurrence of field 2 has the value CAN.

L The last occurrence. For example, (13)F2 = "-" (F1 = F3,L). In this example field 3 is a repeatable field while fields 1 and 2 are not. The statement is that field 2 must be a hyphen if the value of field 1 is the same as the value of the last occurrence of field 3.

P Previous occurrence. For example, (10) P ([5S],P[10] !@). This states that set 10 is prohibited if it was not used in the previous occurrence of the segment beginning with set 5.

N Any occurrence. For example, (5)F4 = [4]F3,N. This means that field 4 of set 5 must equal the value of some occurrence of field 3 of set 4.

T This occurrence. This is used only to modify a repeatable set, segment, field group, or field operand. It means the number of this occurrence. For example, (10)F1 = [10S],T means that field 1 of set 10 must contain a value equal to this occurrence number of the segment beginning with set 10. That is, in the third occurrence of the segment beginning with set 10, the value of field 1 in set 10 must be 3.

A,B,C, Corresponding occurrences. These subscripts are used to indicate corresponding occurrences of different operands when such correspondence is non - trivial. For example, (8)Q([7]@) means that set 8 is required if set 7 occurs. It is considered trivial and therefore unnecessary to specify by subscript that if these sets are in a segment then the domain of the action is within each occurrence of the segment. This concept of domain is discussed more completely in paragraph 4. But consider the following expression:
(6)F2 Q ((F1 = [5]FG,AF1) & ([5]FG,AF2 !@)) In this case, the condition under which field 2 of set 6 is required involves two fields in a repeatable group of fields in a different set. By using the subscript A on both field groups, it is explicitly made clear that the simple condition must use the same (corresponding)

occurrence of the repeating group of fields. The condition might be stated in this way: "...if field 1 of set 6 equals some occurrence of field 1 of set 5 and field 2 in the same occurrence of the repeating group of fields of set 5 is not used".

4. **Domain**

The domain is the range over which the operands of an expression are applied. The simplest and most intuitively obvious of such domains is the MTF itself. For example, (6)Q ([5] @) states that set 6 is required if set 5 is used. This condition refers to the use of set 5 within the same occurrence of the message in which set 6 is being considered for use. The fact that set 5 may have been used in some other occurrence of this message type is irrelevant to the message at hand. The notation of domain is used elsewhere in the standard as, for example, in the occurrence category of fields. In that case, the domain is not the entire MTF occurrence, but only the containing set. That is, an occurrence category of Q does not mean that the field is required in every message, but only that it is required when the set is used.

In a similar manner, the domain for a given set is the message containing it unless the set is in a segment. In this latter case the domain is the occurrence of the containing segment.

Usually, all operands in an expression will be in the same domain. When this is not the case, domains can be modified through the use of subscripts as described in paragraph 3.c. Sometimes it is necessary to specify a particular occurrence that cannot be implied by the rules of domain and subscripts. In such instances a complete path name for an operand can be used. A path name consists of the segment, all the nested segments, set, field group, and field, all appropriately subscripted to whatever level necessary to specify the desired operand. For example, [7S],3[9S],L[12],1F3,N refers to any occurrence of field 3 in the first occurrence of set 12 in the last occurrence of the nested segment beginning with set 9 in the third occurrence of the segment beginning with set 7.

There are several rules of domain governing the application of each operand. These rules are specified in the following sections.

Rule 1. The left operand of a statement is applied to all occurrences of the operand, one at a time, in the message. For example, if the left operand of a statement is (4)F2, each occurrence of set 4 , field 2 will be evaluated according to the rest of the expression. The domain of the left operand is the extent of its containing format occurrence. For example, (3) Q states that set 3 is required. But it is required only in each occurrence of the segment that contains it. If the segment does not occur in the message, set 3 does not occur either. Such a statement shall have a condition associated with it.

Rule 2. The right operand of a statement, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of the statement. For example, given the statement, (8)F1 = [7]F1. If both sets are in the same segment, then within each occurrence of the segment field 1 of set 8 must equal field 1 of set 7. Note that if set 8 is repeatable, then all occurrences of set 8 must have a field 1 that matches field 1 of set 7 within that occurrence of the containing segment. But if set 7 is also repeatable within the segment, the statement becomes meaningless because it is not known which occurrence of set 7 within the segment occurrence is being referenced. If it is intended that field 1 in the first occurrence of set 7 is to be matched by field 1 in the first occurrence of set 8 and so forth through all occurrences of both sets, the subscripts should be used in this manner: (8),AF1 = [7],AF1. The subscripts indicating corresponding occurrences are necessary because the repetitions of set 7 and 8 have not been grouped together within the segment by a nested segment.

Rule 3. The left operand of a condition, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of a statement. For example, given the statement, (8)F4 Q (F2@). If F2 and F4 are in a repeating group of fields, in each occurrence of the group there must be a value for field 4 if there is a value for field 2 in that same occurrence of the group of fields. Now consider the same example but in the case where field 4 is but field 2 is not in the repeating group of fields. In this case a value in field 2 will require that a value occurs in field 4 for all occurrences of the repeating group of fields.

Rule 4. The right operand of a condition, unless modified by a subscript, is limited to those occurrences in direct line of nesting to the left operand of the condition. For example, given the condition, ([8]F2 = F4). The comparison of fields occurs only within the same occurrence of set 8. If, by rule 3, several occurrences of set 8 must be evaluated, they must all evaluate true in order to return true for the condition. Within a given occurrence of set 8, if F4 is repeatable but F2 is not, all of the values of F4 must be equal to F2 in order to evaluate to true. Note, however, that since the repeating group of fields has not been indicated, only those occurrences of the repeating group in which F4 has a value (not a hyphen) will be considered. If it is intended that F4 in all occurrences of the repeating group of fields must equal the value in F2 in order to return a true evaluation, the condition should be written as follows: ([8]F2 = [8]FGF4). By specifying the repeating group explicitly, all occurrences of the repeating group are qualified for use. If it is intended that a true should be returned if F2 matches any occurrence of F4, then the right operand of the condition should be modified by a subscript as follows: ([8]F2 = [8]FG,NF4). In this case, the same result would be obtained by affixing the subscript to the field and not specifying the repeating group of fields. That is, ([8]F2 = F4,N).

5. Syntax of Structured Notation[3]

The BNF variant used in this paragraph uses the following primitives:

- Non-terminal symbols are enclosed in <>.
- Terminal symbols are enclosed in quotes and capitalized.
- Production alternatives are separated by a vertical bar *.
- Non-terminal symbols are separated from their productions by ::=.
- Optional items are enclosed in square brackets [].
- Repeatable items are enclosed in curly brackets {}.
- Comments are enclosed in asterisks ** and are written in italics.

[3] [3] The updated paragraph is documented in Annex G to the Case Law Document

```
                E.g. *This is a comment*
<expression>::=
                <statement>["("<condition>")"]|<statement simple operand>
                <unary statement director>"("<condition>")"|<statement left field id>
                " A "<literal operand>

<statement>::=
                <statement left operand><dyadic operator><right operand>|
                <statement left operand><alternative dyadic operator>
                <alternative right operand>| <statement simple operand>
                <dyadic occurrence count operator><right operand>|
                <statement left field group id><dyadic occurrence count operator>
                <right operand>|<statement left operand>"?<"<remainder operand>

<condition>::=
                <simple condition>|"("<condition>")"|
                <condition><logical operator><condition>

<simple condition>::=
                <square simple operand>[","<corresponding subscript>]
                <unary condition operator>|
                <condition left operand><dyadic operator>
                <compound right operand>|
                <condition left operand><alternative dyadic operator>
                <extended compound right operand>|<square simple operand>
                <dyadic occurrence count operator><right operand>

<condition left operand>::=
                <subscripted square field id>","<subscript>|<square field id>

<statement left operand>::=
                <subscripted statement left field id>","<subscript>|
                <statement left field id>

<compound right operand>::=
                <right operand>[{" & "<right operand>}]|<right operand>
                [{" / "<right operand>}]

<extended compound right operand>::=
                <extended right operand>[{" & "<extended right operand>}]|
                <extended right operand>[{" / "<extended right operand>}]
```

```
<extended right operand>::=
                <right operand>|<alternative right operand>

<right operand>::=
                <subscripted square field id>","<subscript>|<square simple
                operand>[",T"]|<integer>

<alternative right operand>::=
                <literal operand>|<wildcard operand>|<fud operand>

<statement simple operand>::=
                <statement left segment id>|<statement left set id>|
                <statement left field id>

<square simple operand>::=
                <square segment id>|<square set id>|<square field group id>|
                <square field id>


<wildcard operand>::=
                [<literal operand>]{"*"<literal operand>}["*"]|<literal operand>"*"

<fud operand>::=
                "FF"{<digit>}(1,4)"-"{<digit>}(1,3)

<remainder operand>::=
                "R"[<increment><integer>]

<literal operand>::=
                """<literal>"""
                        *This produces a literal in quotation marks.*

<statement left field id>::=
                <statement left field group id>","<subscript>"F"<integer>|
                <subscripted statement left field id>

<square field id>::=
                <square field group id>","<subscript>"F"<integer>|
                <subscripted square field id>

<subscripted statement left field id>::=
                <statement left set id>[","<subscript>]"F"<integer>

<subscripted square field id>::=
                <square set id>[","<subscript>]"F"<integer>|"F"<integer>

<statement left field group id>::=
                <statement left set id>[","<subscript>]"FG"

<square field group id>::=
                <square set id>[","<subscript>]"FG"

<statement left set id>::=
                [<statement left segment id>","<subscript>]"("<integer>")"

<square set id>::=
                [<square segment id>","<subscript>]"["<integer>"]"

<statement left segment id>::=
                [{<statement left segment id>","<subscript>}]"("<integer>"S)"|
                [{<statement left segment id>","<subscript>}]"("<integer>"I)"

<square segment id>::=
                [{<square segment id>","<subscript>}]"["<integer>"S]"|
                [{<square segment id>","<subscript>}]"["<integer>"I]"

<subscript>::=
                "L"|"N"|"P"|<corresponding subscript>|<integer>

<corresponding subscript>::=
                "A"|"B"|"C"

<increment>::=
                " + "|" - "

<logical operator>::=
                " & "|" / "

<unary condition operator>::=
                " @ "|" !@ "

<unary statement director>::=
                " Q "|" P "|" QP "

<dyadic operator>::=
                " > "|" < "|<alternative dyadic operator>

<alternative dyadic operator>::=
                " = "|" != "

<dyadic occurrence count operator>::=
                " @= "|" @> "|" @< "|" @! "

<literal>::=
                {<char>}
```

```
<char>::=
                <digit>|<special char>|" "|"A"|"B"|"C"|....|"X"|"Y"|"Z"
<integer>::=
                {<digit>}
<digit>::=
                "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"
<special char>::=
                "."|","|"-"|"("|")"|"?"
```

ANNEX D

ABBREVIATIONS

ABBREVIATION OF ADATP-3 TERMS

TERM	**ABBREVIATION**
ADatP-3 Central Data Base System	ACDBS
Allied Communications Publication	ACP
Allied Data Publication	ADatP
Allied Data Systems Interoperability Agency	ADSIA
Allied Tactical Publication	ATP
Alphabetic Character	A
Alphanumeric Character	X
Alternative Field Format	AFF
Backus Naur Form	BNF
Conditional (Occurrence Category)	C
Data Entry Identification Field	DE
Field Format	FF
Field Format Index Reference Number	FFIRN
Field Use Designator	FUD
Field Use Designator Number	FUDN
Keyword-Out-of-Context	KWOC
Major NATO Commander	MNC
Mandatory (Occurrence Category)	M
Message Text Format	MTF
NATO Message Text Formatting System	FORMETS
North Atlantic Treaty Organization	NATO
Numeric Character	N
Operationally Determined (Occurrence Category)	O

ABBREVIATION OF ADATP-3 TERMS

TERM	**ABBREVIATION**
Blank Character (Space)	B
Special Character	S
Structural Notation	SN
Structured Language	SL

This page is intentionally blank

GLOSSARY
TERMS AND DEFINITIONS

The following definitions are applicable to this publication.

TERM	DEFINITION
Alphabetic Character	One of the capital letters from A to Z.
Alphanumeric Character	An alphabetic, numeric, special or blank character.
Alternative Content Field	A field offering a choice between two or more specified field formats.
Alternative Field Format	Two or more field formats offered for use in a field position of a set format from which one must be chosen.
Amplification Set	A free text set (Set Format Identifier AMPN) used to insert an explanation or additional information concerning only the preceding set.
Available Character	An alphanumeric character, colon (:) or slant (/).
Blank Character	A character that causes the print or display position to advance one position along the line without producing any graphic character.
Character	A letter, digit or other symbol used as representation of data.
Closing Text	A subdivision of a formatted message text that provides for unique national requirements that cannot be accomodated within the structure of the main message text. When used, it follows the main message text and is introduced by the identifier CLOSTEXT; its structure is not oth-erwise governed by FORMETS.
Column Header	A word or abbreviation, placed at the head of a column of a columnar set, that describes the type of information contained in that column.
Columnar Set	A formatted set having a group of fields that provide an ordered collection of data aligned vertically under a horizontal array of column headers.

TERM	DEFINITION
Component FUD	An elemental FUD used in the make-up of a composite field format.
Composite Field	A field, the content of which represents a chain of linked data items and that is made up of elemental fields structured in accordance with specified field formats.
Composite Field Format	A field format comprising an ordered sequence of elemental field use designators.
Composite FUD	A field use designator comprising an ordered sequence of elemental field use designators.
Conditional	The categorization of the occurrence of those fields, sets and segments whose use (required or prohibited) depends upon some prestated structural feature of the message.
Data Code	One or more alphanumeric characters used to represent a data item.
Data Entry Identification Field	A numeric field used in the first column of a columnar set to achieve correlation with the lines of one or more other columnar sets in the same message.
Data Item	An indivisible unit of information.
Elemental Field	A field for which the field content represents a single data item.
Elemental Field Format	A field format for which data items and associated data codes are specified.
Elemental Field Use Designator	A field use designator associated with an elemental field format.
End-of-Set Marker	Two consecutive slant characters (//) that terminate a set.
Field	A field marker followed by the field content.
Field Content	The field content comprises either a DATA CODE with a FIELD DESCRIPTOR if specified,

TERM	DEFINITION
	or a single hyphen character. BLANK CHARACTERS may be included in columnar sets for justification and positioning.
Field Descriptor	A combination of one to eight alphabetic or numeric characters and a colon (:) which may be used in a field before the data code to cue the reader to the type of information being addressed.
Field Format	A field format is a specification of one or more fields that are to contain similar data.
Field Format Index Reference Number	A unique number assigned to each field format.
Field Format Name	A unique name given to each field format.
Field Format Structure	The designation of the number and permitted types of characters required for the data codes associated with the field format.
Field Marker	A slant symbol (/) that marks the start of each field.
Field Position	The sequence number in which a field format is specified as a component of a set format.
Field Use Designator	A title that uniquely identifies a specific type of field format usage and, when used in a set format, provides context meaning to a field position.
Field Use Designator Number	A number assigned to each field use designator that is unique within a field format.
Field Use Designator Structure	The designation of the number and permitted types of characters required to represent the data codes associated with the field use designator.
Fixed-Length Field Format	A field format that requires an invariable number of characters.
Formatted Message	A message whose text is organized in a

TERM	DEFINITION
	predetermined sequence of standard formatted subdivisions suitable for human and automatic interpretation and processing.
Free Text Field	A field with no prescribed length limitation and allowing all available characters except two successive slant characters. This type of field is used only in the AMPN, NARR, RMKS, and GENTEXT sets.
Free Text Set	A set containing a free text field that allows free form information, usually in natural language, to be included in formatted messages.
General Text Set	A free text set (Set Format Identifier GENTEXT), consisting of two fields. The first field provides a formatted indication of the subject which is addressed in the following field and, where appropriate, its relationship to preceding sets. The second field is a free text field.
Group of Field Formats	Two or more contiguous field formats associated for the purpose of repetition. The group must include the final field format of the set format. In a columnar set format the group comprises all the field formats. For both linear and columnar set formats the group may degenerate to a single field format.
Group of Sets	Two or more contiguous sets.
Header Line	The second line of a columnar set that contains appropriately spaced field markers and column headers.
Introductory Text	A subdivision at the beginning of a formatted message text, that provides for message handling information. Note: It often contains, but is not limited to, the overall security classification for the message, flag words, special handling instructions, subject indicator codes (SIC), and, where appropriate, message section identification. Its structure is not

TERM	DEFINITION
	governed by FORMETS.
Left-Justify	To position data within the space allocation so that the first data character occupies the left-most character position of the field (applies to columnar sets only).
Line	A horizontal array of not more than 69 characters.
Linear Set	A formatted set having one or more fields that provide data in a linear manner.
Main Message Text	A (Main) Message Text is a group of sets that are related and are transmitted together as a message. The message text of a formatted message is governed by FORMETS.
Mandatory	The categorization of the occurrence of those field formats, set formats, and segments that are related to essential information.
Message	Any thought or idea expressed in plain, coded, or encrypted language, prepared in a form suitable for transmission by any form of communication.
Message Text	Short form for the term Main Message Text.
Message Text Format	A specification of a message name, a message identifier, and the identifiers of the set formats of those sets to be included in a message text in the sequential order in which they are to be used. Also included are the designations of any segments that may be used in the message.
Message Text Format Identifier	A unique group of characters assigned to each message text format and used as a key to the structure and information content of that format.
Message Text Format Index Reference Number	A unique identifier assigned to each message text format.
Message Text Format Name	A unique name given to each message text

TERM	DEFINITION
	format.
Narrative Set	A free text set (Set Format Identifier NARR) used to insert an explanation or additional information concerning two or more immediately preceding sets.
Nested Segment	A segment embedded in another segment.
Numeric Character	One of the digits from 0 to 9.
Occurrence Category	The designation assigned to field formats, set formats, and segments that determines how the formats are to be treated in the message text. The designations are mandatory, conditional, or operationally determined.
Operationally Determined	The categorization of the occurrence of those field formats, set formats, and segments related to information whose use is determined solely by operational considerations such as availability of information, environmental considerations, or relevant events and conditions. Their use (inclusion) in preparing a message is at the discretion of the message drafter.
Remarks Set	A free text set (Set Format Identifier RMKS) used to insert an explanation or additional information concerning the entire message. It can only be used by the message drafter and when used must be the final set of the Main Message Text.
Right-Justify	To position data within a space allocation so that the last data character occupies the right-most character position of the field (applies to columnar sets only).
Segment	Two or more contiguous set formats that are related by content and may be repeated as a group.

TERM	DEFINITION
Set	See "Linear Set", "Columnar Set" and "Free Text Set".
Set Format	A specification of a set format name, a set format identifier, the identifiers of the field formats and field use designators of those field formats to be included in a set format, and, for columnar set format, the column headers and the set layout.
Set Format Identifier	A unique group of characters assigned to each set format as a key to the structure and information content of that format. Note 1: Linear and free text set format identifiers comprise three to eight alphabetic characters. Note 2: Columnar set format identifiers comprise one numeric character followed by two to seven alphabetic characters.
Set Format Name	The unique name given to each set format.
Set Position	The sequence number in which a set format is specified as a component of a message text format.
Special Character	One of the characters . , - ( ) ?.
Variable-Length Field Format	A field format that allows use of a variable number of characters within prescribed limits.

LIST OF EFFECTIVE PAGES (LEP)

Effective Pages	Page numbers
CHANGE 3	i to viii
CHANGE 3	1-1 to 1-4
CHANGE 3	2-1 to 2-2
CHANGE 3	3-1 to 3-14
CHANGE 3	4-1 to 4-12
CHANGE 3	5-1 to 5-12
CHANGE 3	6-1 to 6-9
CHANGE 3	A-1 to A-8
CHANGE 3	B-1 to B-15
CHANGE 3	B-1-1 to B-1-4
CHANGE 3	C-1 to C-2
CHANGE 3	C-1-1 to C-1-36
CHANGE 3	C-2-1 to C-2-16
CHANGE 3	D-1 to D-4
CHANGE 3	Glossary-1 to Glossary-9
CHANGE 3	LEP-1 to LEP-2

This page is intentionally blank