Software & Information Industry (SIIA) Financial Information Services Division (FISD)

# FISD's XML Messaging Specification

# "fisdMessage" Version 1.0-beta 16 December 2003

# fisdMessage Reference Guide Appendices



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/

SIIA/FISD Chief Technologist:
James E. Hartley, FISD/SIIA, Denver, U.S.A.

Editor:
James E. Hartley, FISD/SIIA, Denver, U.S.A.





## Abstract

The FISD Message Technical Specification ("fisdMessage") defines the protocol and "on-wire" encoding of statically formed eXtensible Markup Language (XML) documents in a flexible and efficient manner. These appendices provide additional information, including examples, about the specification and its processing.

This document uses the Market Data Definition Language (MDDL) to illustrate examples and demonstrate the functionality. MDDL is a specification based on the XML standard to enable interchange of data necessary to account for, to analyze, and to trade instruments of the world's financial markets.

# Appendices





 

 

 

# FISD's XML Messaging Specification

## Appendix A – fisdMessage Pre-Defined Framework

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# A fisdMessage 1.0-beta Pre-Defined Framework

This section discusses the complete set of files from the release that constitute the minimum framework of fisdMessage. The files may be downloaded from the release website at: http://www.mddl.org/fisdMessage/2003/1.0-beta/.

## A.1 Main Directory

The main directory identifies all message templates and enumerations that are part of an fisdMessage session. The default main directory for the release should NOT be modified in any way (although the default text and integer compressions may be changed).

NOTE: Two directories are identified - the default framework directory and the provider-specific directory. Thus, the provider can make changes to its directory without affecting the framework.

## A.2 Message Templates

The message templates are provided as part of the specification and are used to interpret the packets delivered as part of the datafeed protocol. There is one template for each message. Note that the header is common across all messages as defined by the fisdMessage schema that can be used to validate the message templates.

Message templates are differentiated from content templates only in that they are provided as part of the specification and define the format of the messages used in the protocol. The content templates use a similar syntax but define templates and content that are delivered using the messages of the protocol.

## A.3 Specification Enumerations

The specification defines enumerations which are required to properly process the protocol messages. Although enumerations are delivered as part of the protocol, and fisdMessage protocol recipients are encouraged to programmatically process templates and enumerations, the protocol defined enumerations should not be altered in any way to avoid confusion.

Note that these enumerations MAY be used by the provider in the content templates. If a provider wishes to modify the enumerations, a formal change should be made through the specification owner. Alternatively, a provider may copy an enumeration, change the URI used to identify the enumeration, and carry the enumeration as one of its own.

### A.3.1 messageType

The messageType enumeration is fixed at 64 entries and is universally encoded in 6 bits. No fisdMessages will ever be identified with the value "0" for the messageType. The messages are broken into three separate ranges for processing:

1. Session Establishment messages are defined in the range SMsg_FIRST to SMsg_LAST exclusive of the range markers.
2. Administrative messages are defined in the range AMsg_FIRST to AMsg_LAST exclusive of the range markers.
3. Content messages are defined in the range CMsg_FIRST to CMsg_LAST exclusive of the markers.

An application should use the mnemonics in the enumeration when processing as the sequence and numbering of messages is subject to change with subsequent releases - if



 

the message identifiers are abstracted (rather than hard-coded), applications can be readily adapted to future versions of the specification.

### A.3.2 requestStatus

The requestStatus enumeration is fixed at 256 entries and is encoded in 8 bits. The value of "0" is reserved for successful operations while other results will be indicated by values greater than 0.

### A.3.3 compressionType

The compressionType enumeration is fixed at 256 entries and is encoded in 8 bits. The value of "0" is reserved for "no compression" (or "none") while all other compressions are represented by a value greater than 0. The enumeration is expected to be exhaustive for all possible compressions that are recognized by fisdMessage. Note, however, that the naming conventions for the compressions (discussed in another appendix) allow for more values than are defined in the enumeration. Applications should determine the appropriate compression based on the string equivalence of the enumeration value (rather than on the value itself) because this enumeration is like to change frequently.

### A.3.4 encryptionMethod

TO BE COMPLETED… The encryptionMethod enumeration is fixed at 256 entries and is encoded in 8 bits.

### A.3.5 entryType

The entryType enumeration is fixed at 256 entries and is encoded in 8 bits. The entryType is used in the directory to identify whether an entry is a message template, a content template, or an enumeration. The specification defines message templates while the provider defines the content templates.

### A.3.6 enumerationType

The enumerationType is fixed at 256 entries and is encoded in 8 bits. The enumerationType specifies the format of the raw enumeration as provided. The provider will parse and map the enumeration but will provide the raw enumeration for special application processing by the recipient.

### A.3.7 fieldType

The fieldType is fixed at 256 entries and is encoded in 8 bits. The fieldType identifies the basic (atomic) type of the field (see the appendix on field characteristics).

### A.3.8 fieldUnits

To be completed…



 

# FISD's XML Messaging Specification

## Appendix B – Description of Field Characteristics

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# B Description of Field Characteristics

To be completed…

## B.1 Number

The field number is the unique number within the template for this field. When content messages are sent to update a field, this field number is referenced. Based on the compression scheme used for field updates, the provider should sequence fields such that the most frequently updating fields (referenced in content update messages) should be given smaller numbers. Many systems will assign the smaller field numbers to compound (or "virtual") fields because they are used most frequently.

## B.2 Mnemonic - Field Identifier

The field mnemonic provides a unique identifier for this field within the template. The mnemonic may be a text string, a URI, or an XPath (of the field within the content XML specification) at the discretion of the provider. It is recommended that the mnemonic be unique across all templates except where the similarity is explicit (as is done with fisdMessage templates).

## B.3 Name

The name is the human-readable identifier for the field which may be used in display applications. The default name must be provided in English (at this time). In subsequent releases, multiple names may be provided in other languages with a default language, other than English, specified.

## B.4 Type - Atomic Type

The type is the basic (atomic) type of a field as defined by the enumeration "fieldType" and is currently limited to Boolean, integer, signed integer, string, enumeration, or float.

## B.5 Units

To be completed…

## B.6 Display Units

To be completed…

## B.7 Size

The size specifies the number of bytes that should be allocated for the data entity by the recipient. A value of 0 indicates that the fields is a zero-based string which should be dynamically allocated as necessary.

## B.8 Minimum Value

The minimum value specifies the smallest value that can be assigned to the integer-represented field (which includes all integers, Booleans, and enumerations).

## B.9 Maximum Value

The maximum value specifies the largest value that can be assigned to the integer-represented fields (which includes all integers, Booleans, and enumerations).

## B.10 Entitlement

The entitlement is a 32-bit unsigned value, assigned by the provider, which identifies a particular access or entitlement that the user (or subsequent downstream user) must



 

have in order to access the data value of the field. If the entitlement expression is non-zero then the entitlement expression may specify a formula that uses the entitlement. If the entitlement expression is null then the entitlement is a stand-alone value controlling access to the field. If the entitlement is 0 and the entitlement expression is null then no special accesses are required to view the content.

## B.11 Entitlement Expression

The entitlement expression is a null-terminated string, assigned by the provider, which specifies an expression identifying special accesses or entitlements that the user (or subsequent downstream user) must have in order to access the data value of the field. This expression may be a formula that uses the entitlement value as well. If the entitlement is 0 and the entitlement expression is null then no special accesses are required to view the content.

## B.12 Description

The description is the human-readable description of the field and the value it contains. The default description must be provided in English (at this time). In subsequent releases, multiple descriptions may be provided in other languages with a default language, other than English, specified.

## B.13 Default Compression

The default compression is a string that identifies which compression scheme is normally used for compressing a field. The default compression scheme is used to compress all values sent via a "CMsg Compressed Field Update".

## B.14 Compressions

Any field may be compressed with more than one compression. The legal compressions for a field are identified by the provider. Note that this list will NOT contain the default compression and will NOT contain the standard "none" compression (which may be applied to any field). This list defines the compressions that can be used in a "CMsg Alternate Field Update" message.

## B.15 Dependencies

The dependencies list other fields that coordinate with the field to define a collection of data that should be updated simultaneously - or may be related to one another. Normally, a list of dependencies is used to create a compound field (see subordinates) that will be used to send all of the fields of the dependencies simultaneously.

## B.16 Subordinates

The subordinates identify fields that are collected to define a compound (or "virtual" field). An update containing this field will include the concatenation of all of the referenced subordinates. Note that compound fields are NOT sent as part of an image update and will only be used for updates.

## B.17 Subordinate Formula

The subordinate formula is (currently) an interpretive logic expression that should be applied to the value provided in a compound field update before updating the value of the subordinate field. As an example, the formula may identify scaling to be applied to the value or may identify an explicit value to assign to the subordinate field. See the appendix section on formulas for more information.



 

## B.18 Source

The source is a moniker that a processing system will used to identify where the data comes from within the system configuration – this identifies the datafeed, database, or processing engine providing the data (NOT the organization name).

## B.19 Formula

As with the subordinate formula for subordinate fields, the formula specifies an interpretive logic expression that should be applied to the provided value before updating the actual value of a field.

## B.20 Key Field and Index Bits

If the field is identified as a key field then this field is the key field and may include an enumeration-like mapping for values of the key. The number of index bits configures the receiver to know the approximate number of instances of this template to expect.

## B.21 Frequency of Updates

The frequency of updates is a clue to the provider to assist in assigning the field numbers to fields within the templates. This information is not used by the recipient.



 

# FISD's XML Messaging Specification

## Appendix C – Converting an Image into an Instance

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# C Converting an Image into an Instance

To be completed…

The **fisdMessage** protocol is designed to get updates to a frequently changing XML document from provider to consumer. It does this by separating the raw XML document into "static" and "dynamic" portions. The static sections are transmitted once (during *template* definition) and *updates* to individual fields (defined in the *template*) are delivered as necessary. The *recipient* application can construct a valid XML instance document by applying the fielded values to the *template* in a methodical fashion outlined here.

## C.1 Discussion of Method

The *template* contains a combination of XML from the **fisdMessage** schema as well as the schema of the XML variation that is to be transmitted. One can see the form of the resultant XML instance document by simply removing the **fisdMessage** markup from the *template*. However, this will not generate an exact document.

The *recipient* of **fisdMessage** content intending to create XML documents is required to keep the latest value of each of the core fields transmitted (as an *image* with subsequent *updates*). The values stored correspond directly to a field within the *template* and thus will be represented in the output XML instance document.

The algorithm for performing the transformation is rather straightforward and should not require significant processing time:

- Begin by copying all non-fisdMessage content from the template to the output XML instance document. Use XML guidelines regarding whitespace if this is important.
- When a template field is defined (denoted by the fisdMessage tag **<field>**), obtain the value from the stored list for that field.
- Use the definition of the field (including storage and display types with transformation rules) to convert the stored value to an appropriate string.
- Place the resultant string into the output XML instance document.
- Continue and repeat until all core fields have been processed and the remaining contents of the template have been copied to the output.

## C.2 Special Considerations

*Undefined Values*

During processing of an *image* some fields may contain undefined values. The *template* is constructed such that any XML element that has no content (no attributes and no child elements) should be removed from the resulting instance document. Note that compound and virtual fields are NOT represented in the XML instance document so only the status of the core fields should be examined.



 

# FISD's XML Messaging Specification

## Appendix D – Compression Schemes

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# D Compression and Encoding Schemes

The efficient conveyance of XML encoded data requires compression. In many cases, standard textual compression of documents may be sufficient to meet bandwidth requirements. However, **fisdMessage** is based on the assumption that this is not true for high-volume high-throughput content that is statically formed. Using the *image* with *updates* methodology, **fisdMessage** is used to separate the XML encoding from the raw data fields. These fields can then be manipulated in their "native" binary form and conventional compression schemes can be applied against these token data items.

The following discusses the various compression and encoding schemes that can be applied to **fisdMessage** content. These schemes only apply to the encoding of the native type discussed and are independent of the data being represented. For example, the number "123.45" may be represented as an integer in a complex field (12345 divided by 100) rather than as a floating point number.

## D.1 Scheme "none" – No Compression

When a field is not compressed in any way then the compression is defined as "none" and all fields marked as such will be conveyed using the full number of bytes allowed for that field (in the case of integers, enumerations, floating points, and fixed-length character strings). Note that enumerations will still be passed as integers (rather than the string value equivalent). The number of bits required for the enumeration-based integer is a function of the number of entries defined in the enumeration.

## D.2 Scheme "zlib" – Whole Message/Field Text Compression

Many messages are large and may carry significant amounts of text - especially the enumeration and template messages that contain XML documents. To reduce the size of these large messages, whole message compression can be enabled by setting the "compressed" flag in the message header. By default, the compression method used for text is zlib, which is available via a free license at http://www.zlib.org/.

The AMsg Directory Response defines (in the field "compression") the default text compression for all Administrative and Content messages - EXCEPT the AMsg Directory Response and the AMsg Timestamp. The AMsg Timestamp MUST NOT be compressed and the AMsg Directory Response MAY ONLY be zlib compressed. Similarly, the Session Establishment messages MAY ONLY be zlib compressed. A mechanism for specifying an alternate default compression scheme for most of the Session Establishment messages and the AMsg Directory Response is being investigated.

In some cases, a field may contain enough text that it can be compressed on its own. As with any field, "zlib" may be defined as a viable compression for that field. If this is done, the first 3 bytes (subject to compression as defined in "compressionInt" of the AMsg Directory Response) of the new payload MUST CONTAIN the length of the uncompressed textual string for processing at the receiving location.

## D.3 Text Field Encoding

To be completed…

### D.3.1 Scheme "s[n]" – [n] Byte Fixed Length String

The text field is a fixed length number of UTF-8 bytes as denoted by the qualifier [n]. The string is NOT null-terminated.



 

### D.3.2 Scheme "utf8" – Text Field Unicode Encoding

The default text encoding for all text strings is the Unicode derived "utf8" encoding as is normal with XML documents. This encoding is valuable in that it allows the representation of the entire universe of characters in a standard "string" way - there are no null characters except the terminating null. More can be learned about this encoding at http://www.unicode.org/.

### D.3.3 Scheme "5bitCS1" – Limited Character Set 1

The characters of the field are a null-terminated string formed from a subset of the 32 characters of the set below and are "squashed" together to form a contiguous bit stream with 0 padding at the end as appropriate. This encoding will allow 8 characters from the following set to be conveyed in 5 bytes:

<null>**ABCDEFGHIJKLMNOPQRSTUVWXYZ&./:?**

Limited Character Set 1 encompasses the null character, the 26 capital letters of the ASCII character set in addition to the ampersand "&", the period, ".", the slash "/", the colon ":", and the question mark "?" numbered 0-31 starting from the left.

### D.3.4 Scheme "6bitCS2" – Limited Character Set 2

The characters of the field are a null-terminated string formed from a subset of the 64 characters of the set below and are "squashed" together to form a contiguous bit stream with 0 padding at the end as appropriate. This encoding will allow 8 characters from the following set to be conveyed in 6 bytes:

<null> **!"#$%&'()*+,-.:;<=>?/0123456789@ABCDEFGHIJKLMNOPQRSTUVWXYZ_|~**

Limited Character Set 2 encompasses the null character, the ASCII characters 32 through 90, the carat (^), the underscore (_), the pipe (|), and the tilde (~).

### D.3.5 Scheme "7bit" – ASCII Text Field Compaction

The text of the field is predetermined to be ASCII and thus all characters are truncated to 7 bits and "squashed" together into a contiguous bit stream with 0 padding as appropriate. This encoding will allow 8 ASCII characters to be conveyed in 7 bytes.

## D.4 Date and Time Encoding Schemes

The dateTime string of XML (based on ISO 8601) can take 32 bytes to represent a date and time (for example, 2003-09-09T09:09:09.000000+01:00 to represent microseconds at one hour ahead of GMT). As the **fisdMessage** specification is focused on bandwidth-efficient delivery of XML encoded data, it is necessary to find a better way to represent time. Further, as time can be frequently used throughout a system, a way of specifying a delta from some known time reference is reasonable.

**fisdMessage** defines an 8 byte date and time value as the concatenation of two different fields - 5 bytes to represent the number of seconds since midnight on 01 January 1970 followed by 3 bytes to hold the number of microseconds ($10^{-6}$) elapsed in the referenced second. Both portions should be treated as integers but no compression is defined for these fields - a full timestamp always has 8 bytes.

There is a "heartbeat" defined in the specification - the AMsg Timestamp message - with a full date and time which is normally transmitted frequently by the *sender*. The *sender* may take advantage of this frequent message and calculate deltas from the timestamp when convenient. Further, if multiple **fisdMessage** packets are combined into a single packet (see CMsg Blocked Field Update) then delta times between times in a packet can be used to minimize bandwidth necessary for timestamps.



 

### D.4.1 Scheme "tH[[s]n.x]" – Delta Time from Heartbeat

This timestamp "compression" scheme provides a count of the number of $10^{-x}$ seconds in relationship to the last heartbeat using *n* bits to encode the value. For example, tH4.3 would use 4 bits to represent the number of microseconds ($10^{-3}$) since the last heartbeat. As the delta time may require referencing time *before* the heartbeat, the optional *s* indicator specifies that the delta time is signed. Thus, tHs4.3 would indicate 4 bits of milliseconds treated as a signed number.

### D.4.2 Scheme "tB[[s]n.x]" – Delta Time from Blocking

The CMsg Blocked Field Update allows multiple update messages to be grouped into a single packet with a single timestamp that relates to the whole packet. The "Delta Time from Blocking" scheme allows other times within the packet to be in relationship to the blocked message timestamp (rather than the heartbeat itself). Note that the blocked message timestamp may be in relationship to the heartbeat or may be a full timestamp. The encoding of "Delta Time from Blocking" is identical to that of the "Delta Time from Heartbeat" scheme except for the different time reference.

## D.5 Integer and Enumeration Encoding Schemes

The integer and enumeration encoding schemes are straightforward adaptations of common practices for integral values. The base type for all integral numbers should be assumed to be an integer of at least 4 bytes. The reality of most data is that the integral number can be represented with far fewer bits.

### D.5.1 Scheme "i[[s]n]" – [n] Bit Encoding

With this encoding scheme, *n* bits are used to represent the integer. For example, "i24" specifies that 24 bits (3 Bytes) are necessary to hold the maximum value. The *s* qualified indicates the values is a signed integer. When specifying an integral value, this scheme dictates the maximum number of bits needed to represent the value. This scheme is normally used to define the basic type of a field but smaller representations may be used in some cases.

### D.5.2 Scheme "d[[s]n]" – [n] Bit Delta Encoding

As with the integer encoding scheme defined above, the delta encoding scheme specifies the number of bits used to represent a delta since the last update of that field. Similarly, the *s* qualifier indicates that these bits represent a signed value.

### D.5.3 Scheme "i[s]4[.m]" – Continuation Bit after Nybble [m]

The integer value is encoded with nybbles (4 bit groupings). The upper bit, starting with the first nybble after the *m*th nybble is reserved as a continuation bit. For example "i4.0" indicates that there are more nybbles if the high-order bit in the first nybble is set. Likewise, "i4.1" indicates that the first nybble contains four bits of information to be concatenated with the lower three bits from the following nybbles. If the high-order bit is clear in any of the following nybbles, it is the last nybble to be concatenated.

### D.5.4 Scheme "i[s]8[.m]" – Continuation Bit after Byte [m]

As with the nybble continuation scheme, this scheme identifies groups of 8 bits that are used for the encoding where the high-order bit represents a continuation bit. Similarly, the *s* qualifier denotes a signed value when all of the bits are concatenated.

## D.6 Boolean Encoding Schemes

A Boolean is intended to indicate one of two values and thus only requires 1 bit. However, compound field definitions and template processing may be facilitated by having other compression schemes available.



 

### D.6.1 Scheme "b1" – 1 Bit Boolean
The Boolean is encoded in a single bit.

### D.6.2 Scheme "b4" – 4 Bit Boolean
The Boolean is encoded in 4 bits (a nybble).

### D.6.3 Scheme "b8" – 8 Bit Boolean
The Boolean is encoded in 8 bits (a byte).

## D.7 Float Encoding Schemes
All floating point representations within **`fisdMessage`** are governed by the IEEE 754 specification for floating point numbers.

Although floating point numbers can be transported with **`fisdMessage,`** there are fewer encoding schemes that are appropriate as many so called "real" numbers can be expressed as integers to a sufficient precision multiplied or divided by a power of 10. Thus, a complex field may be defined that allows for a more efficient coding than the standard floating point representations.

### D.7.1 Scheme "f32" – Single Precision
The single precision representation of floating point values is governed by IEEE standard 754.

### D.7.2 Scheme "f64" – Double Precision
The double precision representation of floating point values is governed by IEEE standard 754.



 

# FISD's XML Messaging Specification

## Appendix E – Encryption Framework

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# E Encryption Framework

To be completed… This section will discuss the fisdMessage conventions around the use of AES, DES, MD5, RSA, and other encryption schemes and structures within the protocol. The encryption for a particular session is negotiated will the messages "SMsg Session Notification" and "SMsg Session Notification Response" or is preconfigured as directed by the provider via an out-of-band communication.



 

# FISD's XML Messaging Specification

## Appendix F – Primary Field Handling

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# F Primary Field Handling

To be completed… General guidelines on what to do if a field is updated.

## F.1 Instance Document Index – "The Key Field"

To be completed…

## F.2 Character Strings (and URIs)

To be completed…

## F.3 Date and Time

To be completed…

## F.4 Integers

To be completed…

## F.5 Enumerations

To be completed…

## F.6 Booleans

To be completed…

## F.7 Floats

To be completed…



 

# FISD's XML Messaging Specification

## Appendix G – Suggested Monitoring Parameters

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# G Suggested Monitoring Parameters

To be completed… Guidelines on how to define a template for collecting statistics about throughput and message counts or events.

## G.1 Introduction

To be completed…

## G.2 Recommendations

To be completed…

## G.3 Examples

To be completed…



 

# FISD's XML Messaging Specification

## Appendix H – Complete Examples

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# H Complete Examples

To be completed…

## H.1 Introduction

To be completed…



 

## H.2 Enumerations

To be completed…



 

## H.3 Templates

To be completed…



 

## H.4 Session Establishment Messages

To be completed…



 

### H.4.1 SMsg Service Notification

To be completed…

Template Outline for SMsg Service Notification:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgServiceNotification</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgServiceNotification>
  </SMsgServiceNotification>
</fisdMessage>
```

Example Instance for SMsg Service Notification:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.2 SMsg Service Request

To be completed…

Template Outline for SMsg Service Request:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgServiceRequest</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgServiceRequest>
  </SMsgServiceRequest>
</fisdMessage>
```

Example Instance for SMsg Service Request:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.3 SMsg Service Response

To be completed…

Template Outline for SMsg Service Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgServiceResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgServiceResponse>
  </SMsgServiceResponse>
</fisdMessage>
```

Example Instance for SMsg Service Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.4 SMsg Authentication Request

To be completed…

Template Outline for SMsg Authentication Request:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgAuthenticationRequest</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgAuthenticationRequest>
  </SMsgAuthenticationRequest>
</fisdMessage>
```

Example Instance for SMsg Authentication Request:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.5 SMsg Authentication Response

To be completed…

Template Outline for SMsg Authentication Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgAuthenticationResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgEnumerationResponse>
  </AMsgEnumerationResponse>
</fisdMessage>
```

Example Instance for SMsg Authentication Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.6 SMsg Entry Request

To be completed…

Template Outline for SMsg Entry Request:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgEntryRequest</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgEntryRequest>
  </SMsgEntryRequest>
</fisdMessage>
```

Example Instance for SMsg Entry Request:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.7 SMsg Session Notification

To be completed…

Template Outline for SMsg Session Notification:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgSessionNotification</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgSessionNotification>
  </SMsgSessionNotification>
</fisdMessage>
```

Example Instance for SMsg Session Notification:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.8 SMsg Session Notification Response

To be completed…

Template Outline for SMsg Session Notification Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgSessionNotificationResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgSessionNotificationResponse>
  </SMsgSessionNotificationResponse>
</fisdMessage>
```

Example Instance for SMsg Session Notification Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.9 SMsg Session Termination Request

To be completed…

Template Outline for SMsg Session Termination Request:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgSessionTerminationRequest</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgSessionTerminationRequest>
  </SMsgSessionTerminationRequest>
</fisdMessage>
```

Example Instance for SMsg Session Termination Request:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.10 SMsg Query Instance

To be completed…

Template Outline for SMsg Query Instance:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgQueryInstance</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgQueryInstnace>
  </SMsgQueryInstance>
</fisdMessage>
```

Example Instance for SMsg Query Instance:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.11 SMsg Query Unmark

To be completed…

Template Outline for SMsg Query Unmark:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgQueryUnmark</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgQueryUnmark>
  </SMsgQueryUnmark>
</fisdMessage>
```

Example Instance for SMsg Query Unmark:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.12 SMsg Query All Data

To be completed…

Template Outline for SMsg Query All Data:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgQueryAllData</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgQueryAllData>
  </SMsgQueryAllData>
</fisdMessage>
```

Example Instance for SMsg Query All Data:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.4.13 SMsg Query Other

To be completed…

Template Outline for SMsg Query Other:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>SMsgQueryOther</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <SMsgQueryOther>
  </SMsgQueryOther>
</fisdMessage>
```

Example Instance for SMsg Query Other:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

## H.5 Administrative Messages

To be completed…



 

### H.5.1 AMsg Timestamp

To be completed…

Template Outline for AMsg Timestamp:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgTimestamp</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgTimestamp>
    <timestamp><!-- timestamp --></timestamp>
    <version><!-- version --> </version>
    <sequenceNumber><!-- sequenceNumber --></sequenceNumber>
    <previousTimeDelta><!-- previousTimeDelta --></previousTimeDelta>
  </AMsgTimestamp>
</fisdMessage>
```

Example Instance for AMsg Timestamp:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.2 AMsg Directory Response

To be completed…

Template Outline for AMsg Directory Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgDirectoryResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgDirectoryResponse>
  </AMsgDirectoryResponse>
</fisdMessage>
```

Example Instance for AMsg Directory Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.3 AMsg Enumeration Response

To be completed…

Template Outline for AMsg Enumeration Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgEnumerationResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgEnumerationResponse>
  </AMsgEnumerationResponse>
</fisdMessage>
```

Example Instance for AMsg Enumeration Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.4 AMsg Enumeration Renumber

To be completed…

To be completed…

Template Outline for AMsg Enumeration Renumber:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgEnumerationRenumber</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgEnumerationRenumber>
  </AMsgEnumerationRenumber>
</fisdMessage>
```

Example Instance for AMsg Enumeration Renumber:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.5 AMsg Enumeration Extend

To be completed…

Template Outline for AMsg Enumeration Extend:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgEnumerationExtend</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgEnumerationExtend>
  </AMsgEnumerationExtend>
</fisdMessage>
```

Example Instance for AMsg Enumeration Extend:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.6 AMsg Enumeration Remove

To be completed…

Template Outline for AMsg Enumeration Remove:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgEnumerationRemove</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgEnumerationRemove>
  </AMsgEnumerationRemove>
</fisdMessage>
```

Example Instance for AMsg Enumeration Remove:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.7 AMsg Template Response

To be completed…

Template Outline for AMsg Template Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgTemplateResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgTemplateResponse>
  </AMsgTemplateResponse>
</fisdMessage>
```

Example Instance for AMsg Template Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.8 AMsg Template Remap

To be completed…

Template Outline for AMsg Template Remap:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgTemplateRemap</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgTemplateRemap>
  </AMsgTemplateRemap>
</fisdMessage>
```

Example Instance for AMsg Template Remap:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.9 AMsg Template Expansion

To be completed…

To be completed…

Template Outline for AMsg Template Expansion:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgTemplateExpansion</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgTemplateExpansion>
  </AMsgTemplateExpansion>
</fisdMessage>
```

Example Instance for AMsg Template Expansion:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.10 AMsg Template Remove

To be completed…

Template Outline for AMsg Template Remove:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgTemplateRemove</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgTemplateRemove>
  </AMsgTemplateRemove>
</fisdMessage>
```

Example Instance for AMsg Template Remove:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.11 AMsg Instance Reindex

To be completed…

Template Outline for AMsg Enumeration Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgInstanceReindex</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgInstanceReindex>
  </AMsgInstanceReindex>
</fisdMessage>
```

Example Instance for AMsg Instance Reindex:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.5.12 AMsg Instance Remove

To be completed…

Template Outline for AMsg Instance Remove:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>AMsgInstanceRemove</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <AMsgInstanceRemove>
  </AMsgInstanceRemove>
</fisdMessage>
```

Example Instance for AMsg Instance Remove:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

## H.6 Content Messages

To be completed…



 

### H.6.1 CMsg Image Update

To be completed…

Template Outline for CMsg Image Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgImageUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgImageUpdate>
  </CMsgImageUpdate>
</fisdMessage>
```

Example Instance for CMsg Image Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.2 CMsg Fielded Image Update

To be completed…

Template Outline for CMsg Fielded Image Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgFieldedImageUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgFieldedImageUpdate>
  </CMsgFieldedImageUpdate>
</fisdMessage>
```

Example Instance for CMsg Fielded Image Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.3 CMsg Uncompressed Field Update

To be completed…

Template Outline for CMsg Uncompressed Field Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgUncompressedFieldUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgUncompressedFieldUpdate>
  </CMsgImageUpdate>
</fisdMessage>
```

Example Instance for CMsg Uncompressed Field Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.4 CMsg Compressed Field Update

To be completed…

Template Outline for CMsg Compressed Field Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgCompressedFieldUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgCompressedFieldUpdate>
  </CMsgCompressedFieldUpdate>
</fisdMessage>
```

Example Instance for CMsg Compressed Field Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.5 CMsg Alternate Field Update

To be completed…

Template Outline for CMsg Alternate Field Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgAlternateFieldUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgAlternateFieldUpdate>
  </CMsgAlternateFieldUpdate>
</fisdMessage>
```

Example Instance for CMsg Alternate Field Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.6 CMsg Compressed Range Update

To be completed…

Template Outline for CMsg Compressed Range Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgCompressedRangeUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgCompressedRangeUpdate>
  </CMsgCompressedRangeUpdate>
</fisdMessage>
```

Example Instance for CMsg Compressed Range Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.7 CMsg Undefine Field Update

To be completed…

Template Outline for CMsg Undefine Field Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgUndefineFieldUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgUndefineFieldUpdate>
  </CMsgUndefineFieldUpdate>
</fisdMessage>
```

Example Instance for CMsg Undefine Field Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.8 CMsg Instance Expansion Update

To be completed…

Template Outline for CMsg Instance Expansion Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgInstanceExpansionUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgInstanceExpansionUpdate>
  </CMsgInstanceExpansionUpdate>
</fisdMessage>
```

Example Instance for CMsg Instance Expansion Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.9 CMsg Blocked Field Update

To be completed…

Template Outline for CMsg Blocked Field Update:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgBlockedFieldUpdate</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgBlockedFieldUpdate>
  </CMsgBlockedFieldUpdate>
</fisdMessage>
```

Example Instance for CMsg Blocked Field Update:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

### H.6.10 CMsg Query Response

To be completed…

Template Outline for CMsg Query Response:

```
<fisdMessage version="1.0-beta">
  <header>
    <messageType>CMsgQueryResponse</messageType>
    <encrypted><!-- encrypted --></encrypted>
    <compressed><!-- compressed --></compressed>
    <messageLength><!-- messageLength --></messageLength>
    <fullLength><!-- fullLength --></fullLength>
  </header>
  <CMsgQueryResponse>
  </CMsgQueryResponse>
</fisdMessage>
```

Example Instance for CMsg Query Response:

```
To be completed…
```

The byte stream transmitted for the above message looks like this:

```
Field…
```

Resulting instance document after the above message is distributed:

```
<?xml version="1.0" encoding="utf-8"?>
```



 

# FISD's XML Messaging Specification

## Appendix I – Internet Checksum Algorithm

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# I Internet Checksum Algorithm

To be completed…

Derived from reference in "Computer Networks - A Systems Approach" by Larry L. Peterson and Bruce S. Davie:

```
u_short
chksum_16( u_short *buf, int count )
{
  register u_long sum = 0;

  while( count-- )
  {
    sum += *buf++;
    if( sum & 0xFFFF0000 )
    {
      /* carry occurred,
         so wrap around */
      sum &= 0xFFFF;
      sum++;
    }
  }

  return( ~(sum & 0xFFFF) );
}
```



 

# FISD's XML Messaging Specification

## Appendix J – Field Formula Specificaiton

## Version 1.0-beta 16 December 2003



http://www.mddl.org/ -> http://www.fisd.net/ -> http://www.siia.net/



 

# J Field Formula Specification

To be completed… This section is key to the highest compression levels.

The field and subordinate formulas specify logic that should be applied to a field before it is updated. These formulas can be used to greatly reduce the data that is necessary to send via a field update. Examples of field updates include:

Example 1 **F2: = <value>*5**
This formula (to the right of the colon) will set the value of the current field to whatever value was provided in the update (or in the image) multiplied by 5.

Example 2 **F3: F2= <value>*10**
Assuming field "F3" is properly defined as a compound field (thus has "F2" as a subordinate), this formula will update field "F2" with whatever value is provided after multiplying that value by 10. Note that if field "F2" also has a field level formula (as with Example 1) then the field level formula will be also be applied. Assuming that field "F2" has the field level formula identified and F3 is identified as above, then the result of this instruction will be that F2 will contain the provided value multiplied by 50.

Example 3 **F5: F3= -1**
This instruction will updated field "F3" to the integer value "-1" (after passing the value -1 though the field level formula defined for F3).

Example 4 **F6: F3= -1, F2= <value>*10**
This instruction defines a field "F6" that is the concatenation of two fields - "F3" followed by "F2". Note that because the subordinate formula for F3 sets that field to a specific value, no value for F3 is actually provided in the update. Thus, field "F6" changes the value of two fields although only one is provided in the update.

Example 5 **F7: F3= -1, F2= <value>*10. F1= F1 + F2**
As with the example above, this formula also causes field "F1" to be incremented by the NEW value of field F2. Thus, three fields are actually updated by providing one value.

Example 6 **F8: F3= <undefined>**
This formula will set field "F3" to an undefined value thus preventing it from showing up in an output XML document derived from the template and instance. This has the same effect as sending a "CMsg Undefine Field Update" for field "F3". Note that field level formulas are not applied to undefined values EXCEPT where a field formula tests the condition of the field.

Example 7 **F3: If <value> == <undefined> Then <value> = -1**
This field level formula tests the value of field "F3". If the value is undefined then the field takes on the value -1.



 