Whiteflag Protocol Specification

Status	
version	1
status	DRAFT 6
date	28 FEB 2019

1 Introduction

1.1 Background

Current armed conflicts are highly complex, because of the sheer number of parties involved: regular military forces, armed groups, peacekeeping forces, neutral parties such as journalists and non-governmental human-rights and aid organisations, civilians, refugees etc. Even though parties are opposing forces, or neutral organisations that do not want to show any affiliation, they do require to quickly and directly communicate to one or more other parties involved in the conflict in different situations.

This is not new. The white flag is the original internationally recognized protective sign of truce or ceasefire, and request for negotiation. A white flag signifies to all that an approaching negotiator is unarmed, with an intent to surrender or a desire to communicate.

This standard for a digital white flag protocol, the Whiteflag Protocol, provides a reliable means for both combatant and neutral parties in conflict zones to digitally communicate pre-defined signs and signals using blockchain technology. These sign and signals can also be used to communicate information about natural and man-made disasters, thus creating shared situational awareness beyond conflicts.

All in all, the protocol forms the basis for a neutral and open network, the Whiteflag Network, for trusted real-time messaging between parties in conflicts and disaster response.

1.2 Purpose

The purpose of the Whiteflag Protocol is to provide an open, real-time, trusted communication channel between any or all parties in conflict and disaster zones, without the requirement for a trusted third party or any specific software or system. It is the intention to provide a means of communication to be used that is fast, reliable and accessible by everyone, in addition to internal communications within organisations and in addition to any currently existing but limited communication methods such as flags, signs, registers etc.

The purpose of this document is to describe and define the protocol and the message format of the communication channel for policy makers and specialists to be able to adopt and implement the Whiteflag Protocol.

1.3 Document Structure

Chapter 1 “Introduction” describes this document. Chapter 2 “Overview” defines the scope and gives an overview of the protocol, followed by typical use case examples in Chapter 3 “Use Case Examples”. Chapter 4 “Message Format” provides the detailed messages descriptions, and Chapter 5 “Protocol” the detailed protocol description.

1.4 Used Terminology

The following phrases in this specification must be interpreted as follows:

- ... MUST: indicates that something is mandatory, i.e. an absolute requirement
- ... MUST BE ... IF ...: indicates that something is mandatory if some condition is met
- ... MUST BE ... WHEN ...: indicates something is mandatory when something becomes true
- ... MUST NOT ...: indicates that something is not allowed, i.e. an absolute prohibition
- ... MAY ONLY ... IF ...: indicates something is only allowed if some condition is met
- ... MAY ...: indicates that something is allowed
- ... SHOULD: indicates something is strongly recommended
- ... SHOULD NOT: indicates something is strongly not recommended
- ... NEED NOT ... / ... NOT REQUIRED : indicates that something is not mandatory

An overview of definitions of the most important terms used in this document can be found in Annex D.

1.5 License and Usage

All persons and organisations that contributed to the initial development of the Whiteflag Protocol did so disinterestedly.

The Whiteflag Protocol specification is dedicated to the public domain under the Creative Commons CC0-1.0 Universal Public Domain Dedication statement, meaning that to the extent possible under law, the authors and their organisations have waived all copyright and related or neighboring rights to this work, allowing anyone to copy, modify, distribute and implement the work, even for commercial purposes, all without asking permission.

In no way are the patent or trademark rights of any person affected by this dedication to the public domain, nor are the rights that other persons may have in the work or in how the work is used, such as publicity or privacy rights. Unless expressly stated otherwise, the authors and their organisations make no warranties about the work, and disclaim liability for all uses of the work, to the fullest extent permitted by applicable law. When using or citing the work, you should not imply endorsement by the authors or the affirmer.

WARNING: THE USAGE OF SIGNS AND SIGNALS IN THIS SPECIFICATION IS SUBJECT TO LOCAL AND/OR INTERNATIONAL LAWS. Please take note of the following when implementing or using this standard:

- This standard only specifies an additional means of communication for certain signs and signals, and it does not replace any existing standard, regulation or means of communication, whether mandatory or not, including but not limited to physical signs, radio communications, official registers, etc.
- The usage of protective signs is subject to International Humanitarian Law. Misuse of protective signs is a punishable violation under local and international laws.
- The usage of emergency signals may be subject to various regulations and standards, depending on location and context. The misuse of emergency signals, including misuse of the duress functionality, may be a punishable violation under local and international laws.

1.6 Configuration Management

It is foreseen that this standard requires updates in the near future as a result of validation testing and to ensure compatibility with emerging distributed computing practices, paradigms and standards. Therefore, for the time being, the standard will remain under configuration control of the authors. Requests and comments may be e-mailed to the authors:

- T. Schless: `tschless (at) acm . org`
- M. Kolenbrander: `mkolenbrander (at) acm . org`

In the meantime, a more sustainable process for configuration management of this standard in line with its open character is under consideration.

2 Overview

2.1 Design Principles

The following principles are the basis for the Whiteflag Protocol:

- the protocol is based on blockchain technology, but is "blockchain-agnostic", i.e. independent of any specific blockchain;
- the protocol is as free and open as the underlying blockchain and internet technologies are: anyone can join at any time without permission;
- the working of the protocol does not rely on any third party, i.e. there is no ownership of the network that is created with the protocol and no dependency on specific software or a system;
- the protocol does not have access control, but does provide means of authentication;
- the protocol inherits the data integrity and non-repudiation properties of the underlying blockchain(s);
- the protocol allows to use encryption for message confidentiality;
- the protocol should be compliant with international rules and standards for armed conflicts;
- the protocol is kept as simple as possible, to ensure easy access, easy understanding and easy implementation;
- the message formats are fixed to ensure interoperability and a common understanding between the communicating parties;
- the protocol is extensible to allow functionality to be added with backwards compatibility;
- implementation only requires the use of open standards.

2.2 Protocol Stack and Scope

The Whiteflag Protocol works on top of the internet and on one or more blockchains to create a network for trusted communications that can be used by different applications through application programming interfaces.

The protocol stack comprises the following 6 layers, from bottom to top:

- The *Connection Layers* are to ensure global connectivity. Essentially these are the 7 OSI layers as implemented with the Internet.
- The *Blockchain Layer* ensures connectivity and interaction with specific blockchains, typically through their APIs.
- The *Blockchain Overlay Network Layer* is an abstraction for applications using existing blockchains to validate their transactions. In this case, the Whiteflag Network is created as a blockchain overlay network by Whiteflag applications encapsulating Whiteflag messages in blockchain transactions.
- The *Decentralised Protocol Layer* is a protocol that is not controlled by a single entity. That is exactly what the Whiteflag Protocol and the scope of this specification is.
- The *Application Programming Interface (API) layer* contains the solutions that allow software to interface with the Whiteflag Protocol. These APIs may provide useful additions to the protocol such as handling, tracking and searching of Whiteflag messages.
- The *Application Layer* is the part of the stack that comprises the applications that are actually used by the end-users. These might be existing classical applications such as databases in use by various organisations, or newly developed web applications or smartphone apps that can send, receive, filter, analyse messages.

2.3 References

2.3.1 International Rules & Regulations

2.3.1.1 International Humanitarian Law A. Convention (IV) respecting the Laws and Customs of War on Land and its annex: Regulations concerning the Laws and Customs of War on Land. The Hague, 18 October 1907.

B. Convention (I) for the Amelioration of the Condition of the Wounded and Sick in Armed Forces in the Field. Geneva, 12 August 1949.

C. Protocol additional to the Geneva Conventions of 12 August 1949, and relating to the Adoption of an Additional Distinctive Emblem (Protocol III), 8 December 2005.

D. Convention (III) relative to the Treatment of Prisoners of War. Geneva, 12 August 1949.

E. Convention (IV) relative to the Protection of Civilian Persons in Time of War. Geneva, 12 August 1949.

F. Protocol Additional to the Geneva Conventions of 12 August 1949, and relating to the Protection of Victims of International Armed Conflicts (Protocol I), 8 June 1977.

G. Convention on the Safety of United Nations and Associated Personnel. New York, 9 December 1994.

H. Treaty on the Protection of Artistic and Scientific Institutions and Historic Monuments (Roerich Pact). Washington, 15 April 1935.

I. The 1954 Hague Convention for the Protection of Cultural Property in the Event of Armed Conflict and its two (1954 and 1999) Protocols, United Nations Educational, Scientific and Cultural Organization (UNESCO).

2.3.1.2 International Standards J. IMO IA994E International Code of Signals, 2005 Edition.

K. Disaster Category Classification and peril Terminology for Operational Purposes, Centre for Research on the Epidemiology of Disasters (CRED) and Munich Reinsurance Company (Munich RE), October 2009.

L. Economic Infrastructure Common Reporting Standard Codes, Organisation for Economic Cooperation and Development (OECD).

2.3.2 Technical Standards

2.3.2.1 Communication and Data Format Standards M. RFC 3339, Date and Time on the Internet: Timestamps, July 2002, internet: https://www.ietf.org/rfc/rfc3339.txt

N. RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, January 2005, internet: https://www.ietf.org/rfc/rfc3986.txt

O. RFC 4627, The application/json Media Type for JavaScript Object Notation (JSON), July 2006, internet: https://www.ietf.org/rfc/rfc4627.txt

P. RFC 7515, JSON Web Signature (JWS), May 2015, internet: https://www.ietf.org/rfc/rfc7515.txt

Q. ISO 8601, Date and Time Representation

R. ISO 6709, Standard representation of geographic point location by coordinates

2.3.2.2 Cryptographic Standards S. OpenSSL Implementation of the Elliptic Curve Diffie-Hellman (ECDH) algorithm, internet: https://wiki.openssl.org/index.php?title=Elliptic_Curve_Diffie_Hellman&oldid=1558

T. OpenSSL Implementation of the Advanced Encryption Standard (AES)

U. RFC 5639, ECC Brainpool Standard Curves & Curve Generation, March 2010, internet: https://www.ietf.org/rfc/rfc5639.txt

V. RFC 5869, HMAC-based Extract-and-Expand Key Derivation Function (HKDF), internet: https://www.rfc-editor.org/rfc/rfc5869.txt

2.4 High-level Functional Overview

2.4.1 Blockchain and Communication Functionality

2.4.1.1 General A blockchain is a shared database that maintains a continuously-growing list of ordered records called blocks, each containing a timestamp and a hash-based link to a previous block going all the way back to the first block.

Blockchain networks are open source distributed computing systems with high byzantine fault tolerance: secure by design and inherently resistant to modification of the data; once recorded, the data in a block cannot be altered retroactively.

Therefore, a blockchain can be seen as an open, distributed ledger that can record transactions between parties efficiently and in a verifiable and permanent way. This makes blockchains very suitable for the recording of events and messages.

The Whiteflag Protocol defines the messages for signs and signals used in armed conflicts and for disasters, and it defines how those messages can be sent on, in principle, any blockchain network by encapsulating them in transactions. A messages is recorded in the database when such a transaction (including the encapsulated data) gets included in a new block.

The usage of the Whiteflag Protocol on one or more specific blockchain network(s), establishes what in this specification is called the Whiteflag Messaging Network, or just "the network", and can be seen as what is sometimes called a *Blockchain Overlay Network*.

2.4.1.2 Originator and Account The originator is a specific organisation or person that sends Whiteflag Messages on the Whiteflag Network. The originator's identity is established upon initial entry to the network. An originator may use multiple accounts on the blockchain network. Although an account may use multiple (deterministic) addresses, the usage of more than one address by a single account is not recommended for Whiteflag.

It is only required for an originator to provide identity information before sending messages; it is not required to provide identity information to receive messages, as anyone can observe what is happening on the blockchain.

2.4.1.3 Authentication mechanisms An originator must introduce itself on the Whiteflag Network and provide identity information with one or more initial authentication messages. An initial authentication message contains the self-chosen name of the originator, and a method to verify that the originator's blockchain address is under control by the person or organisation the originator claims he is.

Currently, two basic methods of verification are supported by the protocol.

1. The first method makes use of an internet resource, such as a web site, under control of the originator. The initial authentication message of the originator contains an Uniform Resource Locator (URL) to the internet resource, where the originator posts the corresponding blockchain address, along with its proclaimed name, both signed with a digital signature using the address' secret private key. Anyone can now validate that the blockchain address is under control by the originator by performing a check on the validity of the digitally signed message using the blockchain address' public key. This verification method assumes that the originator is in control of the internet resource specified in the initial authentication message, and can therefore be associated with that internet resource. As a result, the initial trustworthiness of the identity is as strong as the internet resource being used: obviously this differs between an SSL-secured web site of an acclaimed organisation and for instance a pseudonymous social media post. Although an internet site is vulnerable to being compromised and an attacker could alter the authentication data on the internet resource, this will never result in a valid digital signature as long as the private key remains secret.
2. The second method uses a pre-shared token. The originator and a receiver exchange a token, which is not known to anybody. The the originator identifies himself by putting a combined hash of the token and the blockchain account in the initial authentication message. The receiver can verify now that the claimed blockchain address actually belongs to the originator who was in prior possession of the token. Note that the secret token itself is not revealed. The originator may do this multiple times with different tokens, different accounts and/or different receivers, and any combination of those.

The initial authentication can be enhanced by using multiple initial authentication messages, i.e. by combining both verification methods, and also use multiple tokens and URLs. Additionally, the protocol allows other participants to confirm the originator's identity, which is especially useful if those confirming participants have well-established identities.

For blockchains that encourage or require to use a different address for each transaction, deterministic key chains are used to link the authentication message both to the blockchain account as well as to other messages sent by the same originator but with different addresses.

The authentication mechanism is described in detail in 2.4.2.2 Management Messages (initial authentication message) and Joining and Leaving the Whiteflag Network (protocol for initial authentication). After initial authentication, the Whiteflag Protocol utilizes the authentication mechanism of the underlying blockchain.

As means for digital identification on the internet are still evolving, it is anticipated that additional methods will be added in future versions.

2.4.1.4 Communications Security The Whiteflag Protocol leverages blockchain technology to ensure data integrity, non-repudiation and historical continuity. The underlying blockchain(s) ensure(s) that any message ever sent will be verifiable, unalterable and available to anybody, even to participants that start to use the protocol later in time.

Blockchain networks using the proof-of-work consensus model (Nakamoto consensus) timestamp transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work.

The timestamp proves that the data must have existed at the time in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it. The hash chain of blocks provides an unchangeable historical record, which is verifiable by anyone.

Nodes can leave and rejoin a blockchain network and thus the Whiteflag Network at will, accepting the proof-of-work chain as validation of what happened while they were gone. Upon rejoining, missing blocks are downloaded, verified and added to the nodes' local copy of the database until it has caught up with the network and is in sync (again).

2.4.1.5 Data Security The Whiteflag Protocol the protocol provides optional message confidentiality using an AES based encryption scheme to encrypt the message contents. This is described in detail in 4.1.4 Encryption.

The encryption scheme allow for both Diffie-Hellman negotiated keys and pre-shared keys. The Whiteflag Protocol specifies the Diffie-Hellman key exchange, but does not specify how pre-shared encryption keys should be managed or distributed. It is between the originator and recipient to agree on a secure key exchange procedure.

2.4.2 Message Functionality

2.4.2.1 Functional Messages The following Functional Messages are specified in this standard. These messages are to send a sign or signal as defined in international standards, and to provide additional information.

Code	Name	Description
`P`	`ProtectiveSign`	Sign to mark objects under the protection of international law
`E`	`EmergencySignal`	Signal to send an emergency signal when in need of assistance
`D`	`DangerSign`	Sign to mark a location or area of imminent danger, e.g. an area under attack, land mines, disaster, etc.
`S`	`StatusSignal`	Signal to provide the status of an object, or specifically for persons: give a proof of life
`I`	`InfrastructureSign`	Sign to mark critical infrastructure, e.g. roads, utilities, water treatment, hospitals, power plants etc.
`M`	`MissionSignal`	Signal to provide information on activities undertaken during a mission
`Q`	`RequestSignal`	Signal to perform requests to other parties
`R`	`Resource`	Message to point to an internet resource
`F`	`FreeText`	Message to send a free text string

A specific Functional Message has a single meaning. For example, a Danger Sign is just that: a Danger Sign, which does not imply that the originator is in danger or requires assistance. In such a case, the originator must send an emergency signal.

However, a Functional Message may be related to an earlier Functional Message by referencing it. In this way, sequences of message can be created, e.g. for moving objects (by updating the coordinates of a Protective Sign), or requesting assistance for an object under attack (by sending an Emergency Signal referring to a Status Signal referring to a Danger Sign). Message sequencing is described in detail in 5.4.2. Message Sequences.

2.4.2.2 Management Messages A number of management messages are defined to provide additional means to verify the validity of the blockchain address and related account, to support cryptographic functions and to test connectivity and functionality.

Code	Name	Description
`A`	`Authentication`	Message introducing the sender on the network with the sender's authentication data
`K`	`Crypto`	Message for management of keys and parameters of cryptographic functions
`T`	`Test`	Message that can be used for testing Whiteflag functionality by applications

2.4.2.3 Duress Indicator The Whiteflag Protocol provides duress functionality, i.e. an indicator that a message has been sent under (threat of) violence, or other pressure and should therefore be interpreted with caution.

This functionality is especially useful for implementations of the protocol in devices used by workers and operators in the field, where there is a higher risk of hijacking and kidnapping resulting in possible misuse of the Whiteflag communication from an address of that account.

This standard only specifies how the duress indication is to be implemented in the protocol and how it should be interpreted. The way the Duress Indicator will be triggered by a Whiteflag capable application or device, is outside the scope of this standard, because this can be implemented in many different ways to suit different scenarios and requirements.

2.4.3 Spatial information

Signs and signals about structures and areas may contain spatial information. The Whiteflag protocol uses the EPSG:4326/WGS84 geodetic coordinate system. The protocol does (currently) not allow to provide height and altitude information, and therefore can only be used to provide an object location on the earth's surface.

Structures and areas can be represented using a limited number of geometric shapes: circles, rectangles and isosceles triangles, and any combination of these.

The real-world structure or area must be fully contained by the geometric shapes representing it. The centre of a geometric shape is determined by the location specified in the Whiteflag message. The orientation of the shape is the angle between a virtual north oriented vector, and the first specified dimension of the shape: this is one of the sides for a rectangle, and the height for an isosceles triangle; a circle does not have an orientation.

2.5 Minimum Implementation

When implementing this standard, it is not required to implement it in its entirety. However, any implementation of the Whiteflag Protocol that *sends* messages on the network, must implement at a minimum the following functionality:

- the `A` message with at least one authentication option;
- the possibility to Recall, Update and Discontinue (Reference Codes `1`, `2` and `4`) any implemented sign or signal, if such Reference Type is valid i.a.w. Reference options.

Also, for both senders and receivers, it is required that any part of this standard that is implemented complies with this standard.

2.6 Implementation Considerations

This paragraph addresses a number of aspects that are technically not part of the standard, but require consideration or are good practices for an interoperable and secure implementation of the Whiteflag protocol.

2.6.1 Trustworthiness of information

The Whiteflag Protocol provides a trusted communication channel in the sense that the originator is authenticated, that messages cannot not be altered or disappear once recorded, all without the requirement of a trusted third party.

However, it is important to understand that a trusted communication channel does not by itself guarantee that the information in a message is true. Upon reception of a Whiteflag message, it is up to the receiver to assess the credibility of the information sent.

Nevertheless, the characteristics of the protocol do help in evaluating the credibility of information:

- The protocol ensures instant verification of the originator. Therefore, if a source is a priori considered reliable by the recipient (which will be often the case for NGOs, journalists, UN peacekeepers, UNMAS, civil authorities, etc.), the information can be considered to have a high level of credibility.
- The protocol allows information to be confirmed by another message of a different originator, and therefore facilitates confirmation of the same information by multiple sources.
- The protocol provides duress functionality;

- Since all information is persistent, this allows for evaluating the reliability of a source over time by cross-checking facts in retrospect.

A recipient may use the following levels of reliability of a source and credibility of information:

Reliability of Source	Credibility of Information
A. Completely reliable	1. Confirmed by other sources
B. Usually reliable	2. Probably true
C. Fairly reliable	3. Possibly true
D. Not usually reliable	4. Doubtful
E. Unreliable	5. Improbable
F. Reliability cannot be judged	6. Truth cannot be judged

Also note that the protocol does not require all messages to be processed by a recipient. A recipient may choose to ignore unknown addresses and accounts, to blacklist unreliable addresses and accounts, to filter messages about an object outside a certain area of interest, create custom perspectives, merge information with other sources, etc. etc.

2.6.2 Account and Private Key Protection

The protocol does not define any specific setup of the blockchain account and private key protection. There are however several important guidelines to consider.

1. Third party hosts should not be used to create accounts on a blockchain, because the identity of the sender should be directly linked to the account. Using a third party to create an account creates attack vectors by which an account could be compromised.
2. Public blockchains provide free, open source clients that facilitate the creation of accounts without any third party involved. Also, it is very possible to create specific Whiteflag software that also manages the associated blockchain accounts and cryptographic keys itself. In general, it is good practice to protect your account by implementing security measures on machines and devices under your own control or supervision.
3. Third party hosts should not be used to store private keys. Storing the secret key outside your control or under supervision creates attack vectors by which an account could be compromised.
4. To protect private keys from theft, hardware wallets, secure processing units and cold wallets should be used when possible.
5. To protect an account from being compromised, multi-factor authentication as well as multi-signature accounts may be used.
6. If account and/or related addresses need to be permanently de-activated, simply delete the private key(s), and, if possible, send an `A(4)` message for every `A(0)` sent before.

2.6.3 Encryption and Key Exchange

The protocol provides the option to encrypt a message for confidentiality. When using encryption, a number of considerations should be taken into account:

1. Even if messages are encrypted, information may leak through usage patterns. Therefore, it is advised not to use the same blockchain addressed and the same proof-of-identity for both encrypted and unencrypted messages.
2. The security of encryption relies largely on the actual implementations.
3. All messages are persistent on the blockchain and it should be assumed that at some point in the future messages can and will be decrypted. Therefore, only use dedicated keys, and only encrypt information that is time-sensitive, i.e. information that will not be valid or relevant anymore over time, such as the status or the location of a moving object.

2.6.4 Processing and Storage of Messages

On the Whiteflag Network, messages must be formatted and processed as indicated in this standard. Additionally, this standard defines how messages and message sequences should be semantically interpreted.

To help implementing and validating Whiteflag Messages, Annex B provides a JSON schema that describes the messages, which might be of beneficial use for application developers.

However, strictly speaking, this schema is outside the scope of this standard: firstly because the Whiteflag Messages sent on the network are not JSON-formatted, and secondly because this standard does deliberately not specify how messages are to be processed and stored by applications (an application might very well use XML instead of JSON). Nevertheless, it is recommended to use a JSON format compliant with this schema for open APIs.

2.7 Note on the notation of Messages and Message Sequences

For documentation and software development purposes, messages codes and message sequences might be written as follows:

- a message may be referred to by its message code, e.g. `T` for a test message;
- a specific sign or signal may be indicated by the combination of message and subject code, e.g. `E01` for a distress signal;
- a reference indicator may be written in between brackets after the message code, e.g. `A(2)` for an update to an earlier `A` message;
- an encrypted message is written between square brackets, e.g. `[Q(0)14]`;
- the originator may be indicated behind the message code and reference indicator, e.g. `A(0)X` is an original authentication message from X;
- the symbol < is used as a pointer in the message sequence; the symbol is repeated to indicate to which message is referred, e.g. << means two messages before.

For example, a message from originator X for relating an area under attack to an existing protective sign from a hospital H may be written as `P31(0)H < D10(5)X`. When the attack ends and the status of the hospital is provided, the full sequence will read: `P31(0)H < D10(5)X < D10(4)X <<< S23(5)H`.

3 Use Case Examples

(*moved to Annex F*)

4 Message Format

4.1 Message Structure

4.1.1 Representations

A Whiteflag message can have different representations at different levels of the protocol stack:

- at the Blockchain Overlay Network Layer (i.e. the level at which the Whiteflag Network is established) the message is represented as a compressed binary string, and optionally encrypted;
- at the Decentralised Protocol Layer, a message is represented as an uncompressed and unencrypted concatenated character string comprised of the message fields;
- at the API layer, a message and its message fields can have any appropriate representation using a structured language, such as JSON or XML;

The standard describes the uncompressed and unencrypted concatenated character string at the Decentralised Protocol Layer, and how these messages are compressed into a binary string, and optionally encrypted.

4.1.2 Encoding

An uncompressed and unencrypted Whiteflag Message consists of:

- the Message Header (described in 4.2 Message Header), which is the same for all messages and consists of 7 fields with a total length of 71 bytes;
- the Message Body (described in 4.3 Message Body), of which the fields and number of bytes per field differ between message types, and is therefore variable in length.

The individual bytes of an uncompressed and unencrypted messages are encoded using a single 7-bit/1-byte UTF-8 code unit, allowing the usage of UTF-8/Unicode code points `U+0000` through `U+007F`, which are identical to the first 128 characters of the ASCII-set.

Depending on the usage of the field, not all characters/code points are allowed for each field. The field descriptions of the message header and body in 4.2 Message Header and in 4.3. Message Body, use the following characters to indicate which subset of characters is valid:

- `c` indicates any valid 1 byte UTF-8/Unicode character, i.e. UTF-8 code points `U+0000` through `U+007F`;
- `a` indicates that any alpha-numeric character (0-9, A-Z, a-z), i.e. UTF-8 code points `U+0030` through `U+0039`, `U+0041` through `U+005A`, and `U+0061` through `U+007A`;
- `b` indicates that only a binary character (0-1) is the be used, i.e. UTF-8 code points `U+0030` trough `U+0031`;
- `d` indicates that only a decimal character (0-9) is to be used, i.e. UTF-8 code points `U+0030` trough `U+0039`;
- `x` indicates that only a hexadecimal character (0-F) is to be used, i.e. UTF-8 code points `U+0030` trough `U+0039`, and `U+0041` trough `U+0046`.

Instead of `d`, the letters `y`, `m`, `d`, `h`, `m`, and `s` are used in some field descriptions in and Message Body to indicate digits representing years, minutes, days, degrees, hours, minutes and seconds respectively. Syntactically they are the same as a decimal character indicated with a `d`, and therefore encoded and compressed in the same way, but semantically they are different.

Capitals used in the field format descriptions of the message header and body in 4.2 Message Header and in 4.3 Message Body, including capitals of the above, indicate that such a character is to be used literally.

Before the message is embedded in a blockchain transaction, the message is compressed and optionally encrypted as described below.

4.1.3 Compression

Whiteflag messages must be compressed to allow usage on blockchains with limited space for custom information, and to improve optional encryption. Compression of the message is done as follows:

- all fixed characters are omitted, except for the Prefix field;
- binary characters (`b`), binary indicators and number signs are compressed to 1 bit;

- decimal and hexadecimal characters (`d` and `h`) are compressed to a half-byte (4 bits) unsigned binary coded decimal/hexadecimal.

The descriptions of the message header and body in 4.2 Message Header and in 4.3 Message Body, show how the compression must be applied to each field.

4.1.4 Encryption

The third field of the message is an indicator for which encryption is being used. This standard currently only provides limited support for encryption, but allows implementation specific encryption mechanisms.

If encryption is used, the whole message must be encrypted, except for the `Prefix`, `Version` and `EncryptionIndicator`, which must never be encrypted.

Note that, depending on the encryption algorithm used, the resulting length of the message may differ from the unencrypted message length, which may complicate the embedding of the message in a transaction on certain blockchains.

4.1.5 Embedding

To finally send the message, after compression and encryption, the message is embedded in a blockchain transaction. How the message is embedded in the transaction depends on the blockchain and is specified in Annex A.

4.2 Message Header

4.2.1 Generic Message Header

All messages use the same generic message header, specified below.

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0-1	2	`Prefix`	Identifies the message as a whiteflag message	`WF`	2x 8-bit UTF-8
2	1	`Version`	Indicates which version of the standard is used to generate the message	`a`	1x 8-bit UTF-8
3	1	`EncryptionIndicator`	Indicates if and which encryption is used	`a`	1x 8-bit UTF-8
4	1	`DuressIndicator`	Indicates whether the message was sent under force or threat	`b`	1x 1-bit binary
5	1	`MessageCode`	Indicates the type of message	`a`	1x 8-bit UTF-8
6	1	`ReferenceIndicator`	Indicates how this message relates to an earlier message	`x`	1x 4-bit unsigned binary coded hexadecimal
7-70	64	`ReferencedMessageID`	Blockchain dependent reference to a related earlier message	`xxxxxxxxxx` `...` `xxxxxxxxxx`	64x 4-bit unsigned binary coded hexadecimal

4.2.1.1 Generic Message Header Fields

4.2.1.2 Prefix Field The `Prefix` field is a fixed value field to easily identify Whiteflag Messages on the blockchain. The filed must contain two 1-byte UTF-8 encoded characters `WF`.

4.2.1.3 Version Field The `Version` field indicates which version of the standard is used as the specification for the message. For implementations based on this version of the specification, the field must be a 1-byte UTF-8 encoded

character 1.

4.2.1.4 Encryption Indicator Field The `EncryptionIndicator` field specifies how if and how a message is encrypted for confidentiality. The contents of the field must be a 1-byte UTF-8 encoded alpha-numeric character i.a.w. the following table.

Code	Encryption	Usage
0	No encryption	No encryption is used
1	`aes-256-ctr-ecdh`	Message is encrypted using 256-bit key AES counter mode with an ECDH negotiated key
2	`aes-256-ctr-psk`	Message is encrypted using 256-bit key AES counter mode with a pre-shared key
3-9	(reserved)	Must not be used
(other)	(private use)	Encrypted with a method indicated by the value, but specific to the application or originator

4.2.1.5 Duress Indicator Field The `DuressIndicator` field value must be `0`, unless the sign or signal was sent under threat or force. In that case the DuressIndicator field must be `1`.

See Duress Indicator for further considerations on the duress functionality.

4.2.1.6 Message Code Field The `MessageCode` field specifies the message type. The contents of the field must be a 1 byte UTF-8 encoded alpha-numeric character value corresponding with a message type defined in this specification.

A message with an invalid, i.e. undefined, message code, must be ignored.

4.2.1.7 Reference Indicator Field The `ReferenceIndicator` field specifies how a message relates to an earlier message. The contents of the field must be a 1-byte UTF-8 encoded value from the following table.

Value	Reference Type	Meaning	Usage
0	`Original`	This message provides new information and is not related to an earlier message.	May be used by any account. Must be used if no other reference type is used.
1	`Recall`	The originator recalls the referenced message; the referenced message must be ignored.	May only be used by the same account of the referenced message. The information in this message must repeat the referenced message.
2	`Update`	This message provides updated information and replaces the information in the referenced message.	May only be used by the same account of the referenced message. The information that did not change must be the same as in the referenced message.
3	`Add`	This message adds additional information to the referenced message, i.e. ties two messages together.	May only be used by the same account of the referenced message.
4	`Discontinue`	The originator discontinues the referenced message, i.e. the information or action is no longer valid.	May only be used by the same account of the referenced message. The information in this message must repeat the referenced message.
5	`Relate`	The information in this message directly relates to the information or action in the referenced message.	May be used by any account, and may reference messages from any other account.
6	`Confirm`	The originator confirms the information or action in the referenced message.	May be used by any account of a different originator than of the referenced message. The information in this message must repeat the referenced message.
7	`Acknowledge`	The originator acknowledges the receipt of the referenced message, but the information itself is not confirmed.	May be used by any account of a different originator than of the referenced message. The information in this message must repeat the referenced message.

Value	Reference Type	Meaning	Usage
8	`Comply`	The originator will comply with and/or act according to the referenced message, but the information itself is not confirmed.	May be used by any account of a different originator than of the referenced message. The information in this message must repeat the referenced message.
9	`Reject`	The originator does not agree (any more) with the information or action in the referenced message.	May be used by any account of a different originator than of the referenced message. The information in this message must repeat the referenced message.

A message with a Reference Indicator that does not comply with these rules must be ignored. However, a recipient may use non-compliant messages to evaluate the reliability of the originator.

By referencing other messages, message sequences can be created. However, not every message type and reference type may reference every other message type and reference type. The rules for referencing and functionality of message sequences are specified in detail in 5.4 Referencing.

4.2.1.8 Referenced Message Field The `ReferencedMessage` field is a 64-byte UTF-8 encoded string containing a unique hexadecimal identifier of the referenced message. This allows for referencing a 256-bit hash. Since other messages can only be identified using the transaction identifier of the underlying blockchain, the identifier depends on the underlying blockchain. If the underlying blockchain uses a hash larger than 256 bits, the first 256 bits of the hash should be used. The transaction identifiers of the most common blockchains are shown in Annex A.

If the `ReferenceIndicator` field is `0`, the `ReferencedMessage` field must be ignored by the receiver. This allows the field to be filled with a random 64-byte UTF-8 encoded hexadecimal string, e.g. to prevent a large part of the message to be all zeros when encrypting it.

4.3 Message Body

4.3.1 Functional Messages: Signs & Signals

4.3.1.1 Signs & Signals Message Fields The message body of functional messages representing a sign (message types `P`, `D` and `I`) or a signal (message types `E`, `S`, `M` and `Q`), must contain the following fields:

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0- 70	71	Message Header	See Generic Message Header Fields		
71-72	2	`SubjectCode`	Indicates the sign/signal type with the value defined in Subject Code Field	`xx`	2x 4-bit unsigned binary coded hexadecimal
73-92	15	`DateTime`	Indicates when the sign/signal is valid, using an ISO 8601/ RFC 3339 timestamp	`YYYY-MM-DDThh:mm:ssZ`	14x 4-bit unsigned binary coded decimal
93-102	10	`Duration`	Indicates how long the sign/signal will be valid, using the ISO 8601 format	`PddDhhHmmM`	6x 4-bit unsigned binary coded decimal
103-104	2	`ObjectType`	Specifies the type of object the sign/signal refers to	`xx`	2x 4-bit unsigned binary coded hexadecimal
105-113	9	`ObjectLatitude`	Specifies the object location in decimal degrees latitude i.a.w. ISO 6709	`\_dd.ddddd`	1x sign bit + 7x 4-bit unsigned binary coded decimal

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
114-123	10	ObjectLongitude	Specifies the object location in decimal degrees longitude i.a.w. ISO 6709	_ddd.ddddd	1x sign bit + 8x 4-bit unsigned binary coded decimal
124-127	4	ObjectSizeDim1	Specifies the size of the object's first dimension in meters	nnnn	4x 4-bit unsigned binary coded decimal
128-131	4	ObjectSizeDim2	Specifies the size of the object's second dimension in meters	nnnn	4x 4-bit unsigned binary coded decimal
132-134	3	ObjectOrientation	Specifies the object's orientation in degrees	nnn	3x 4-bit unsigned binary coded decimal

In addition, the message body of message type Q may be extended with the following fields:

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
135-136	2	ObjectType1	Specifies the type of object for which the request is made	xx	2x 4-bit unsigned binary coded hexadecimal
137-138	2	ObjectType1Quant	Specifies the number of objects for which the request is made	dd	2x 4-bit unsigned binary coded decimal

These fields may be repeated for additional objects for which the request is made, e.g. bytes 139-142 may contain additional ObjectType2 and ObjectType2Quant fields, etc.

4.3.1.2 Subject Code Field The SubjectCode field defines the actual sign or signal, and must be 2-byte UTF-8 encoded value from the following tables with subject codes for each sign/signal message type below. Common official and/or visual equivalents for the defined signs and signals are shown in Annex E.

Message Code	Usage	Subject Code	Usage / Meaning
P	Protective Sign	00	Unspecified Protective Sign
		01	White flag - Unspecified intent
		02	White flag - Request for negotiation
		03	White flag - Surrender
		04-0F	(reserved)
		10	Unspecified Red Cross/Crescent/Crystal
		11	Red Cross
		12	Red Crescent
		13	Red Crystal
		14-1F	(reserved for use by the IFRC)
		20	United Nations
		21-2F	(reserved for use by the UN)
		30	Unspecified Protected Zone
		31	Hospital and Safety Zone
		32	Prisoner of War Camp
		33	Internment Camp
		34-3F	(reserved)
		40	Civil Defence

Message Code	Usage	Subject Code	Usage / Meaning
		41-4F	(reserved)
		50	Unspecified Cultural Property
		51	Monuments and Cultural Institutions
		52	Cultural Property
		53	Cultural Property under Special Protection
		54–5F	(reserved)
		60	Works and Installations containing Dangerous Forces
		61-6F	(reserved)
		70-FF	(reserved)

4.3.1.2.1 Protective Signs References: *P01-03*: i.a.w. Convention (IV) respecting the Laws and Customs of War on Land and its annex: Regulations concerning the Laws and Customs of War on Land. The Hague, 18 October 1907 *P11-12*: Convention (I) for the Amelioration of the Condition of the Wounded and Sick in Armed Forces in the Field. Geneva, 12 August 1949. *P13*: Protocol additional to the Geneva Conventions of 12 August 1949, and relating to the Adoption of an Additional Distinctive Emblem (Protocol III), 8 December 2005. *P20*: Convention on the Safety of United Nations and Associated Personnel. New York, 9 December 1994. *P31*: Convention (IV) relative to the Protection of Civilian Persons in Time of War. Geneva, 12 August 1949 *P32-33*: Convention (III) relative to the Treatment of Prisoners of War. Geneva, 12 August 1949. *P51*: Treaty on the Protection of Artistic and Scientific Institutions and Historic Monuments (Roerich Pact). Washington, 15 April 1935 *P52-53*: The 1954 Hague Convention for the Protection of Cultural Property in the Event of Armed Conflict and its two (1954 and 1999) Protocols, United Nations Educational, Scientific and Cultural Organization (UNESCO) *P60*: Protocol Additional to the Geneva Conventions of 12 August 1949, and relating to the Protection of Victims of International Armed Conflicts (Protocol I), 8 June 1977.

Message Code	Usage	Subject Code	Usage / Meaning
E	Emergency Signal	00	Unspecified Emergency
		01	General Distress signal
		02	General Urgency signal
		03-0F	(reserved)
		10	Urgent Unspecified Assistance Required
		11	Urgent Medical Assistance Required
		12	Urgent Fire & Rescue Assistance Required
		13	Urgent Law Enforcement Assistance Required
		14	Urgent Disaster Response Required
		15-1F	(reserved)
		20-FF	(reserved)

4.3.1.2.2 Emergency Signals

Message Code	Usage	Subject Code	Usage / Meaning
D	Danger Sign	00	Unspecified Danger Area
		10	War/Conflict/Terrorism - Unspecified Attack
		11	War/Conflict/Terrorism - Direct Ground Attack
		12	War/Conflict/Terrorism - Mortar/Artillery Attack
		13	War/Conflict/Terrorism - Ground Bomb Attack
		14	War/Conflict/Terrorism - Air Attack
		15	War/Conflict/Terrorism - Complex Attack
		16-19	(reserved)
		1A	War/Conflict/Terrorism - Chemical Attack
		1B	War/Conflict/Terrorism - Biological Attack
		1C	War/Conflict/Terrorism - Nuclear Attack
		1D-1F	(reserved)
		20	Unspecified Hazardous Conflict Area
		21	Hazardous Conflict Area - Mine Field

Message Code	Usage	Subject Code	Usage / Meaning
		22	Hazardous Conflict Area - UXOs
		23	Hazardous Conflict Area - Chemical Contamination
		24	Hazardous Conflict Area - Biological Contamination
		25	Hazardous Conflict Area - Radiological Contamination
		26	Hazardous Conflict Area - Nuclear Contamination
		27	Hazardous Conflict Area - Surface-to-Air Threat
		28-2F	(reserved)
		30	Unspecified Transport Accident
		31	Transport Accident - Air
		32	Transport Accident - Rail
		33	Transport Accident - Road
		34	Transport Accident - Boat
		35-3F	(reserved)
		40	Unspecified Industrial Accident
		41	Industrial Accident - Accident release (other than attack)
		42	Industrial Accident - Explosion (other than attack)
		43	Industrial Accident - Chemical explosion (other than attack)
		44	Industrial Accident - Nuclear explosion/Radiation (other than attack)
		45	Industrial Accident - Mine explosion
		46	Industrial Accident - Pollution
		47	Industrial Accident - Acid rain
		48	Industrial Accident - Atmosphere pollution
		49-4F	(reserved)
		50	Complex/Man-made Hazard - Complex Emergency
		51	Complex/Man-made Hazard - Uncontrolled Banditry
		52	Complex/Man-made Hazard - Uncontrolled Uprising
		53	Complex/Man-made Hazard - Displaced Population
		54	Complex/Man-made Hazard - Famine and Food Insecurity
		55-5F	(reserved)
		60-9F	(reserved)
		A0	Unspecified Geophysical Disaster Area
		A1	Geophysical Disaster Area - Earthquake
		A2	Geophysical Disaster Area - Earthquake w/ Tsunami
		A3	Geophysical Disaster Area - Volcano
		A4	Geophysical Disaster Area - Dry Mass Movement - Rockfall
		A5	Geophysical Disaster Area - Dry Mass Movement - Landslide
		A6	Geophysical Disaster Area - Dry Mass Movement - Avalanche
		A7	Geophysical Disaster Area - Dry Mass Movement - Subsidence
		A8-AF	(reserved)
		B0	Unspecified Meteorological Disaster Area
		B1	Meteorological Disaster Area - Unspecified Storm
		B2	Meteorological Disaster Area - Storm - Tropical Cyclone
		B3	Meteorological Disaster Area - Storm - Extra Tropical Cyclone
		B4	Meteorological Disaster Area - Local Storm
		B5-BF	(reserved)
		C0	Unspecified Hydrological Disaster Area
		C1	Hydrological Disaster Area - General Flood
		C2	Hydrological Disaster Area - Flash Flood
		C3	Hydrological Disaster Area - Storm Surge
		C4	Hydrological Disaster Area - Wet Mass Movement - Rockfall
		C5	Hydrological Disaster Area - Wet Mass Movement - Landslide
		C6	Hydrological Disaster Area - Wet Mass Movement - Avalanche
		C7	Hydrological Disaster Area - Wet Mass Movement - Subsidence
		C8-CF	(reserved)
		D0	Unspecified Climatological Disaster Area
		D1	Climatological Disaster Area - Extreme Temp - Heat Wave
		D2	Climatological Disaster Area - Extreme Temp - Cold Wave
		D3	Climatological Disaster Area - Extreme Winter Condition
		D4	Climatological Disaster Area - Drought

Message Code	Usage	Subject Code	Usage / Meaning
		D5	Climatological Disaster Area - Wildfire - Forest Fire
		D6	Climatological Disaster Area - Wildfire - Land Fire
		D7-DF	(reserved)
		E0	Unspecified Biological Disaster Area
		E1	Biological Disaster Area - Unspecified Epidemic
		E2	Biological Disaster Area - Viral Infectious Disease Epidemic
		E3	Biological Disaster Area - Bacterial Infectious Disease Epidemic
		E4	Biological Disaster Area - Parasitic Infectious Disease Epidemic
		E5	Biological Disaster Area - Fungal Infectious Disease Epidemic
		E6	Biological Disaster Area - Prion Infectious Disease Epidemic
		E7	Biological Disaster Area - Insect Infestation
		E8	Biological Disaster Area - Animal Stampede
		E9-EF	(reserved)
		F0-FF	(private use, i.e. not standardized)

4.3.1.2.3 Danger & Disaster Signs

Message Code	Usage	Subject Code	Usage / Meaning
S	Status Signal	00	Unspecified positive status
		01-0F	(reserved)
		10	Proof of Life without further details
		11	Proof of Life and Situation Normal
		12	Proof of Life and Contact Requested
		13	Proof of Life but Assistance Required
		14-1F	(reserved)
		20	Object status unknown
		21	Object functional without further details
		22	Object functional and in good order
		23	Object functional but damaged
		24	Object not functional without further details
		25	Object not functional and damaged
		26	Object damaged without further details
		28	Object destroyed
		29-2F	(reserved)
		30-EF	(reserved)
		F0-FF	(private use, i.e. not standardized)

4.3.1.2.4 Status Signals

4.3.1.2.5 Infrastructure Signs

Message Code	Usage	Subject Code	Usage / Meaning
I	Infrastructure	00	Unspecified infrastructure
		01-0F	(reserved)
		10	Transportation - Unspecified transportation infrastructure
		11	Transportation - Road or Highway
		12	Transportation - Mass Transit System
		13	Transportation - Railway
		14	Transportation - Canal or Navigable Waterway
		15	Transportation - Seaport
		16	Transportation - Lighthouse
		17	Transportation - Airport
		18-1F	(reserved)
		20	Energy - Unspecified energy infrastructure

Message Code	Usage	Subject Code	Usage / Meaning
		21	Energy - Electrical transmission/distribution
		22	Energy - Natural gas pipeline
		23	Energy - Petroleum pipeline
		24	Energy - Oil-fired power plant
		25	Energy - Gas-fired power plant
		26	Energy - Coal-fired power plant
		27	Energy - Nuclear power plant
		28	Energy - Hydro-electric power plant
		29	Energy - Geothermal energy installation
		2A	Energy - Solar energy installation
		2B	Energy - Wind energy installation
		2C	Energy - Ocean power installation
		2D	Energy - Biomass installation
		2E-2F	(reserved)
		30	Water management - Unspecified water facility
		31	Water management - Drinking water distribution pipeline
		32	Water management - Drinking water storage reservoir
		33	Water management - Drinking water treatment facility
		34	Water management - Waste water collection/disposal facility
		35	Water management - Drainage system
		36	Water management - Major irrigation system
		37	Water management - Major flood control system
		38	Water management - Coastal water management facility
		39-3F	(reserved)
		40	Communication - Unspecified communication infrastructure
		41	Communication - Postal service
		42	Communication - Telephone land line
		43	Communication - Telephone exchange system
		44	Communication - Mobile phone network system
		45	Communication - Television transmission station
		46	Communication - Radio transmission station
		47	Communication - Internet backbone
		48	Communication - Internet core router
		49	Communication - Undersea cable
		4A-4F	(reserved)
		50	Public service - Unspecified public service
		51	Public service - Hospital
		52	Public service - School
		53	Public service - Police station
		54	Public service - Fire station
		55-5F	(reserved)
		60	Solid waste management - Unspecified solid waste management facility
		61	Solid waste management - Municipal garbage and recyclables collection
		62	Solid waste management - Solid waste landfill
		63	Solid waste management - Solid waste incinerator
		64	Solid waste management - Materials recovery facility
		65	Solid waste management - Hazardous waste disposal facility
		66-6F	(reserved)
		70-EF	(reserved)
		F0-FF	(private use, i.e. not standardized)

4.3.1.2.6 Mission Signals

Message Code	Usage	Subject Code	Usage / Meaning
M	Mission	00	Unspecified Mission
		01-0F	(reserved)

Message Code	Usage	Subject Code	Usage / Meaning
		10	Unspecified Humanitarian Mission
		11-1F	(reserved)
		20	Unspecified Mission under Humanitarian Law
		21	Humanitarian Law - Monitoring
		22	Humanitarian Law - Visiting Prisoners of War and Detained Persons
		23	Humanitarian Law - Aid for Military Personnel - Health Care for wounded, sick or shipwrecked personnel
		24	Humanitarian Law - Aid for Civilian Population - General Aid
		25	Humanitarian Law - Aid for Civilian Population - Food Distribution
		26	Humanitarian Law - Aid for Civilian Population - Health Care
		27	Humanitarian Law - Aid for Civilian Population - Shelter
		28	Humanitarian Law - Aid for Civilian Population - Restoring Family Links
		29	Humanitarian Law - Aid for Civilian Population - Infrastructure Restoration
		2A	Humanitarian Law - Support to Local Government - Advisory
		2B	Humanitarian Law - Support Mission - Administrative
		2C	Humanitarian Law - Support Mission - Logistics
		2D-2F	(reserved)
		30	Unspecified Disaster / Emergency Response Operation
		31-3F	(reserved)
		40	Unspecified Special Political Mission
		41	Special Political Mission - Envoy
		42	Special Political Mission - Sanction panel
		43	Special Political Mission - Monitoring group
		44	Special Political Mission - Field-based mission
		41-4F	(reserved)
		50	Unspecified Diplomatic Mission
		51-5F	(reserved)
		60	Unspecified Law Enforcement Operation
		61-6F	(reserved)
		70	Unspecified Peace Operation
		71	Peace Operation - Conflict prevention
		72	Peace Operation - Peacemaking
		73	Peace Operation - Peacekeeping
		74	Peace Operation - Peacebuilding
		75	Peace Operation - Peace enforcement
		76-7F	(reserved)
		80	Unspecified Military Operation
		81-8F	(reserved)
		90-EF	(reserved)
		F0-FF	(private use, i.e. not standardized)

4.3.1.2.7 Request Signals

Message Code	Usage	Subject Code	Usage / Meaning
Q	Requests	00-0F	(reserved)
		10	Request for Cease Fire
		11-1F	(reserved)
		20	Request for Area Access
		21-2F	(reserved)

Message Code	Usage	Subject Code	Usage / Meaning
		30-EF	(reserved)
		F0-FF	(private use, i.e. not standardized)

4.3.1.3 DateTime Field The `DateTime` field must contain the moment in time using from when the sign or signal is valid, using Coordinated Universal Time (UTC).

This timestamp need not to be identical to the timestamp of the blockchain transaction of the message: the sign or signal may be valid from an earlier or future point in time and may therefore be earlier or later than the timestamp of the blockchain transaction.

Based on ISO 8601 and RFC 3339, the DateTime field must be formatted as follows: `YYYY-MM-DDThh:mm:ssZ`, where:

- `yyyy` are 4 digits for the calendar year
- `mm` are 2 digits for the number of the month; valid values are: `01-12`
- `dd` are 2 digits for the day of the month using; valid values are: `01-31`
- `hh` are 2 digits for the hour of day; valid values are: `00-23`
- `mm` are 2 digits for the minutes; valid values are: `00-59`
- `ss` are 2 digits for the seconds; valid values are: `00-60` (including a possible leap second)
- `Z` is literally the character "Z", indicating UTC.

A message with an invalid DateTime field must be ignored.

4.3.1.4 Duration Field The `Duration` field may contain the time-period for how long the sign or signal is valid counted from the date and time in the `DateTime` field. A sign or signal may be valid for a definite period with a maximum of 99 days 23 hours and 59 minutes, or a minimum of 01 minute, or for an indefinite period.

Based on ISO 8601, the Duration field must be formatted as follows: `PnnDnnHnnM` (i.e. regular expression `\^P\[0-9\]{2}D\[0-9\]{2}H\[0-9\]{2}M\$`), where:

- `P` is literally the character "P", indicating that the string represents a period
- `nn` are 2 digits for the number of days; valid values are: `00-99`
- `D` is literally the character "D", indicating the previous digits represent the number of days
- `nn` are 2 digits for the number of hours; valid values are: `00-23`
- `H` is literally the character "H", indicating the previous digits represents the number of hours
- `nn` are 2 digits for the number of days; valid values are: `00-59`
- `M` is literally the character "M", indicating the previous digits represents the number of minutes

For a sign/signal to be valid for an indefinite period, the duration must be set to 0 as follows: `P00D00H00M`.

A message with an invalid Duration field must be ignored.

4.3.1.5 Object Type Field The `ObjectType` field contains a hexadecimal code to indicate to what sort of object the sign or signal applies to. This information is optional and must be set to 00 if not used. However, it is encouraged to use it, because it might be valuable for the receiver(s) of the message.

Code	Object Type	Usage
00	Not provided	Object type is unknown
01-0F	(reserved)	
10	Unspecified group of people	Unspecified number of humans together
11	Person	An individual human
12	Small group of people	2- 10 humans together
13	Medium group of people	10- 100 humans together
14	Large group of people	100- 1000 humans together

Code	Object Type	Usage
15	Enormous group of people	1000-10000 humans together
12-1F	(reserved)	
20	Unspecified Area	An area is a part of the earths surface, either on land or at sea.
21	Circular Area	
22	Rectangle Area	
23	Triangle Area	
24-2F	(reserved)	
30	Unspecified Structure	Immovable human made individual structures that are fixed to the ground, such as buildings, installations, historic ruins, etc. For non-fixed human made structures or a clustering of structures, such as camps, an area object must be used.
31	Circular Structure	
32	Rectangle Structure	
33	Triangle Structure	
34-3F	(reserved)	
40	Unspecified Land Vehicle	An object used for transport over land
41-4F	(reserved)	
50	Unspecified Water Vehicle	An object used for transport on water
51-5F	(reserved)	
60	Unspecified Air Vehicle	An object for transport in the air
61-6F	(reserved)	
70	Unspecified Space Vehicle	An object for transport in space
71-7F	(reserved)	
80	Unspecified goods	Movable materiel, goods, cargo, supplies, etc.
80-8F	(reserved)	
90-EF	(reserved)	
F0-FF	(private use)	Private use, i.e. not standardized

The table below shows which object types may be used for which message subjects, and which combinations types are not allowed.

Subject Code & Object Code	10-1F Persons	20-2F Areas	30-3F Structures	40-7F Vehicles	80-8F Goods
P01-P0F White Flag Protective Signs	allowed	allowed	allowed	allowed	X
P10-P1F Red Cr* Protective Signs	allowed	allowed	allowed	allowed	allowed
P20-P2F UN Protective Signs	allowed	allowed	allowed	allowed	allowed
P30-P3F Protected Zones	X	allowed	allowed	X	X
P40-P4F Civil Defence Protective Signs	allowed	allowed	allowed	allowed	allowed
P50-P5F Cultural Property Protective Signs	X	allowed	allowed	allowed	allowed
P60-P6F Dangerous Forces Protective Signs	X	allowed	allowed	X	allowed
E00-EFF Emergency Signals	allowed	X	allowed	allowed	X
D00-D1F Attack Danger Signs	X	allowed	allowed	allowed	allowed
D20-D2F Danger Area Signs	X	allowed	X	X	X
D30-D4F Accident Danger Signs	X	allowed	allowed	allowed	X
D50-D5F Uncontrolled Danger Signs	X	allowed	X	X	X
DA0-DFF Disasters	X	allowed	X	X	X
S00-S0F Generic Status Signals	allowed	allowed	allowed	allowed	allowed
S10-S1F Proof-of-life Status Signals	allowed	X	X	X	X

Subject Code & Object Code	10-1F Persons	20-2F Areas	30-3F Structures	40-7F Vehicles	80-8F Goods
S20-S2F Object Status Signals	X	X	allowed	allowed	allowed
I00-IEF Infrastructures	X	allowed	allowed	X	X
M00-MEF Missions	allowed	allowed	allowed	allowed	allowed
Q10-Q2F Requests	X	allowed	X	X	X

4.3.1.6 Object Location Fields The ObjectLatitude and ObjectLongitude fields specify the location of *the centre of the object on, or projected on, the earth's surface* in decimal degrees latitude and decimal degrees longitude, using the WGS84 datum (i.e. EPSG:4326). The vertical location (height or depth) cannot be provided.

Based on ISO 6709 - Standard representation of geographic point location by coordinates, the ObjectLatitude fields must be formatted as follows: _dd.ddddd (i.e. regular expression \^\[+-\]\[0-9\]{2}\\.\[0-9\]{5}\$), where:

- The digits represent the north-south longitude; valid values are: -90.00000 to +90.00000
- _ is either + or - to denote north (+) or south (-).

Based on ISO 6709 - Standard representation of geographic point location by coordinates, the ObjectLongitude fields must be formatted as follows: _ddd.ddddd (i.e. regular expression \^\[+-\]\[0-9\]{3}\\.\[0-9\]{5}\$), where:

- The digits represent the east-west latitude; valid values are: -180.00000 to +180.00000
- _ is either + or - to denote east (+) or west (-).

The object location must be specified for Protective Signs and Danger Signs (Message Codes P and D).

The object location may be used for Emergency Signals and Status Signals (Message Codes E and S), e.g. for security reasons. If the object location is omitted, the coordinates must be set to the following value: +99.99999+999.99999.

The object location may not be omitted in a Status Signal (Message Code S) referencing a Protective Sign or a Danger Sign (Message Code P or D).

4.3.1.7 Object Size Fields The ObjectSizeDim1 and ObjectSizeDim2 fields may be used to specify the horizontal surface dimensions of areas and structures in meters. Valid values are: 0001 to 9999. The value of a dimension that is omitted, must be set to 0000.

For (groups of) persons (Object Codes 10-1F), the Object Size may be provided. The Object Size represents the area in which the person(s) is/are located. The area must be defined as small as possible.

For shaped areas and structures (Object Codes 21-2F and 31-3F), the Object Size must be provided. For unspecified areas and structures (Object Codes 20 and 30) the dimensions are by definition unknown, and must be omitted. For all other objects, the Object Size must be omitted.

For circle shaped areas and structures (Object Codes 21 and 31), the ObjectSizeDim1 field represents the radius; the ObjectSizeDim2 field has no meaning for circle shaped areas and structures and must be omitted.

For rectangle shaped areas and structures (Object Codes 22 and 32), the ObjectSizeDim1 field represents the length of the side for which the orientation is specified with the Object Orientation Field; the ObjectSizeDim2 field represents the other side.

Triangle shaped areas and structures (Object Codes 23 and 33) are isosceles triangles, with the ObjectSizeDim1 field representing the triangle height and the ObjectSizeDim2 represents the base.

4.3.1.8 Object Orientation Field The ObjectOrientation field may be used to specify the orientation of the object in decimal degrees, where 000 means North.

Valid values are: 000 to 359. If the object's orientation is omitted, the field value must be set to 999.

If an Object Size is specified using the ObjectSizeDim1 and ObjectSizeDim2 fields, then the ObjectOrientation field must be specified as well. If the Object Size is not specified, then the Object Orientation must be omitted.

4.3.1.9 Object Request Fields The object request fields ObjectType* and ObjectType*Quant may only be used in Q messages and must be used together, with the object type field before the quantity field.

These fields may be repeated up to seven (7) times, giving a maximum of eight (8) objects with their respective quantity.

The ObjectType* field contains a hexadecimal code to indicate what sort of object the request applies to. The values correspond with those of the ObjectType field as specified in 4.3.1.5, with the following restrictions:

- when referring to persons, the exact number of person must be specified and therefore the 12-1F may not be used;
- unmovable objects, i.e. 20-2F areas and 30-3F structures,may not be used.

Note that in accordance with 4.3.1.5, the object code of a Q message itself must represent an area, i.e. code 20-2F.

Valid values for an ObjectType*Quant field are 00 to 99. If the quantity of an object is unknown, then the respective ObjectType*Quant field must be 00.

4.3.2 Functional Messages: Resource

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0-70	71	Message Header	See Generic Message Header Fields		
71	1	ResourceMethod	Indicates the mechanism for pointing to a resource	x	1x 4-bit unsigned binary coded hexadecimal
72-111*	40*	ResourceData	Provides the data required to find the resource	cccccccccc ... cccccccccc	40x 8-bit UTF-8

4.3.2.1 Resource Message Fields

4.3.2.2 Resource Method Field The ResourceMethod field defines the mechanism for pointing to a resource. The field must be 1-byte UTF-8 encoded hexadecimal character. Currently only one resource method has been defined:

Code	Resource Method	Usage
0	(reserved)	Must not be used
1	InternetResource	Reference to an internet resource
2-9	(reserved)	Reserved for future resource referencing mechanisms
A-F	(private use)	Private use, i.e. not standardized

4.3.2.3 Resource Data Field The content of the ResourceData field depends on the resource method:

- If the ResourceMethod method indicates a reference to an internet resource (resource method 1), then the ResourceData field must contain a valid URL.

The ResourceData field may be longer than 40 bytes, if allowed by underlying blockchain. If the length of the ResourceData field is insufficient to provide the complete URL, an additional Reference Message may be sent with the rest of the URL using reference code 3.

4.3.3 Functional Messages: Free Text

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compress
0-70	71	Message Header	See Generic Message Header Fields		
72-111*	40*	Text	Free text	cccccccccc ... cccccccccc	40x 8-bit

4.3.3.1 Free Text Message Fields

4.3.3.2 Text Field The `Text` field can be used to send 40x 1-byte UTF-8 encoded text, typically to provide additional information about a sign or signal, using reference code 3 or 5.

The `Text` field may be longer than 40 bytes, if allowed by underlying blockchain. If the length of the `Text` field is insufficient to provide all text, an additional text message may be sent using reference code 3.

4.3.4 Management Messages: Authentication

4.3.4.1 Authentication Message Fields The message body of management messages for (initial) authentication (message type `A`), must contain the following fields:

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0-70	71	Message Header	See Generic Message Header Fields		
71	1	`VerificationMethod`	Indicates the authentication mechanism	`x`	1x 4-bit unsigned binary coded hexadecimal
72-111*	40*	`VerificationData`	Provides the data required for authentication	`ccccccccccc ... ccccccccccc`	25x 8-bit UTF-8

4.3.4.2 Verification Method Field The `VerificationMethod` field defines the mechanism for authentication of the blockchain address and related account, and must be a 1-byte UTF-8 encoded hexadecimal character from the following table with codes for each verification method:

Code	Verification Method	Usage
`0`	(reserved)	Must not be used
`1`	`InternetResource`	Authentication through an internet resource
`2`	`SharedToken`	Authentication by a secret pre-shared token
3-9	(reserved)	Reserved for future authentication mechanisms
`A-F`	(private use)	Private use, i.e. not standardized

4.3.4.3 Verification Data Field The content of the `VerificationData` field depends on the verification method:

- If the `InternetResource` method is specified (verification method 1), then the VerificationData field must contain a valid URL;
- If the `SharedToken` method is specified (verification method 2), then the VerificationData field must contain the verification token.

The `VerificationData` field may be longer than 40 bytes, if allowed by underlying blockchain. If the length of the `VerificationData` field is insufficient to provide the complete URL or verification token, an additional Authentication Message may be sent with the rest of the URL or verification token using reference code 3.

4.3.5 Management Messages: Cryptographic Support

4.3.5.1 Cryptographic Support Message Fields The message body of management messages for cryptographic support (message type `K`), must contain the following fields:

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0-70	71	Message Header	See Generic Message Header Fields		

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
71-72	2	`CryptoDataType`	Indicates the type of data in this message	`xx`	2x 4-bit unsigned binary coded hexadecimal
73-144*	80*	`CryptoData`	Contains the cryptographic data	`xxxxxxxxxx` ... `xxxxxxxxxx`	128x 4-bit unsigned binary coded hexadecimal

4.3.5.2 Cryptographic Data Type Field The `CryptoDataType` field defines the type of cryptographic key data to be exchanged, and must be a 2-byte UTF-8 encoded hexadecimal character from the following table with codes for each key type:

Code	Crypto Data Type	`CryptoData` field usage
00	(reserved)	Must not be used; reserved for future use
01	`HDExtPubKey`	Serialised hierarchical deterministic extended public key
02-09	(reserved)	Must not be used; reserved for future use
0A	`ECDHPubKey`	Elliptic Curve Diffie–Hellman public key exchange
0B-0F	(reserved)	Must not be used; reserved for future use
10	(reserved)	Must not be used; reserved for future use
11	`InitVector`	Initialisation Vector for encryption method 1
12-1F	(reserved)	Must not be used; reserved for future use
20	(reserved)	Must not be used; reserved for future use
21	`InitVector`	Initialisation Vector for encryption method 2
22-2F	(reserved)	Must not be used; reserved for future use
20-99	(reserved)	Must not be used; reserved for future encryption features
A0-FF	(private use)	Private use, i.e. not standardized

4.3.5.3 Cryptographic Data Field The content of the `CryptoData` field is determined by the `CryptoDataType` field value as follows.

- When the data type is `HDExtPubKey` (code `01`), the data field must contain the serialised extended public key used to derive the deterministic public keys (and addresses).
- When the data type is `ECDHPubKey` (code `0A`), the data field must contain a compressed elliptic curve public key for a Diffie-Hellman key exchange.
- When the data type is `InitVector` (code `11` or `21`), the data field must contain the 128-bit initialisation vector required to decrypt a message encrypted with the algorithm indicated with encryption indicator code `1` and `2` respectively.

The `CryptoData` field may be longer than 64 bytes if allowed by underlying blockchain. If the length of the `CryptoData` field is insufficient to provide the full data, an additional Cryptographic Support Messages may be sent with the rest of the key data using reference code `3`.

Usage of the Cryptographic Support Message and the data types is detailed in 5.2 Cryptographic Support Functions.

4.3.6 Management Messages: Test

4.3.6.1 Test Message Fields The message body of test messages has an identical set of fields as signs & signals messages have, which allows to test sign and signals. A test message must contain the following fields:

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
0-70	71	Message Header	See Generic Message Header Fields		
71	1	`PseudoMessageCode`	Indicates the message type that is tested	a	1x 8-bit UTF-8

Byte Index	Byte Length	Field	Usage	Uncompressed Encoding	Compressed Encoding
72		Test Message Body	See Message Body		

4.3.5.2 Pseudo Message Code The `PseudoMessageCode` field indicates which sign/signal message is tested. The contents of the field must be a 1-byte UTF-8 encoded alpha-numeric character value corresponding with one of the message types defined in 2.4.2 Message Functionality.

4.3.5.3 Other Test Message Fields All other test message fields are identical in use as the equivalent fields sign/signal message, as described in 4.3 Message Body, with the difference that the fields are shifted 1 byte / 8 bits.

The field names must be preceded by `Test` when used in a test message.

5 Protocol

5.1 Joining and Leaving the Whiteflag Network

5.1.1 Initial Authentication

It is not required to perform initial authentication and provide identity information to receive messages.

To send Whiteflag Messages on the network, the originator must have at least one blockchain account and address. An originator may use multiple accounts, and an account may use multiple addresses.

Each account should be identified by sending an `A(0)` initial authentication message, before sending any other message. Any message sent by an account before that account has sent an `A(0)` messages, may be considered unauthenticated by recipients.

If an account uses multiple addresses, the `A(0)` message must be sent using the address corresponding with public key from which all other addresses can be deterministically derived. For recipients to be able to derive those address, a `K(3)2` message referencing the `A(0)` message must be sent, containing the chain code for child key derivation.

5.1.2 Validating Authentication Information

5.1.2.1 Method 1: URL Validation The URL contained in the VerificationData field an `A1` message must point to web resource. This allows the blockchain account to be linked to a web site, social media account, or any other web resource that identifies the originator. Consequently, the identification is as strong as the web resource used.

The URL should use the `https:` scheme to be able to ensure the validity of the resource.

At the URL, a flattened JSON Web Signature JSON serialization (JWS-JS) formatted object i.a.w. RFC 7515 - JSON Web Signature (JWS) must be found. This object contains a JSON object with authentication information together with a single digital signature.

The digital signature is used for authentication, non-reputation and data integrity. In other words: it ensures that only the owner of the web resource controls the blockchain account used to send the `A` message. The digital signature must be created using the key pair of the associated blockchain account, i.e. the account from which the `A1` message has been sent.

The non-serialised payload with authentication information must be a JSON object itself and must at least have the following properties: the originator's blockchain address, the originator's name, and the same URL as in the A message, as follows:

```
"protected": {
    "addr": "...",
    "orgname": "...",
    "url": "...",
    "extpubkey": "..."
}
```

in which:

- `addr` is the same blockchain address used to send the corresponding `A1` message and of which the corresponding private key is used to create the signature;
- `orgname` is the name of the originator, which can be chosen freely;
- `url` is exactly the same URL as in the `VerificationData` field the corresponding `A1` message.

If the originator uses deterministic keys and addresses, the payload must also contain information to derive the public keys:

- `extpubkey` is the serialised extended parent public key from which the child public keys and addresses used by this originator can be derived.

The serialisation format of an extended public key is blockchain specific. Although supported, deterministic keys should normally not be used.

An example for usage of JWS for authentication is included in Annex C.

5.1.2.2 Method 2: Shared Token Validation The option to use a secret token for authentication allows the issuer of the token to authenticate the originator's blockchain account when the token is revealed in an A2 message. The secret token may be pre-shared or generated from a shared secret:

1. The token may be a secret piece of either arbitrary data or some (encrypted) meaningful data provided to the originator.
2. The token may also be derived from an ECDH shared secret as described in 5.2.3 Key and Token Derivation.

The secret token must not be used directly in a single A2(0) message. Instead, the authentication data sent in the A2(0) message must be derived from the secret token using the HKDF function defined in RFC 5869. The procedure is described in detail in the Key and Token Derivation paragraph.

An originator may use multiple A2(0) messages with tokens from different issuers or ECDH counterparts.

The issuer or ECDH counterparts may send A2(6) or A2(7) messages to confirm or acknowledge the claim, but also may choose not to do so to prevent others to record metadata on communications between parties.

5.1.3 Additional Authentication Information

Multiple tokens, URL verifications, or combinations thereof, may be sent for stronger authentication by sending multiple A(0) messages.

The authentication information in an A(0) or A(2) message may be updated with an A(2) message.

An A(3) message must be used when providing longer URLs or tokens than a single message A(0) or A(2) message can contain in its VerificationData field.

5.1.4 Leaving the Network

An account may leave the network with an A(4) message referencing, and thus discontinuing, each A(0) message that has been sent earlier. Any message sent by an account after that account has discontinued all its A(0) messages, should be considered unauthenticated.

5.2 Cryptographic Support Functions

5.2.1 Hierarchical Deterministic Accounts and Addresses

It may not be assumed that every recipient is able to link deterministically derived addresses to the master public key of the account in order to authenticate the originator.

5.2.2 Key Agreement

The protocol supports cryptographic key exchange using Elliptic Curve Diffie-Hellman (ECDH), which is an Elliptic Curve variant of the standard Diffie-Hellman algorithm. This well known algorithm allows two parties, that do not have any prior knowledge of each other, to agree on a shared secret using an open communication channel. This shared secret may then be used, for example, to derive a secret key for the encryption of messages.

The OpenSSL implementation of ECDH is the reference implementation for for Elliptic Curve Diffie-Hellman key agreement with the Whiteflag Protocol. The elliptic curve parameters that must be used for Whiteflag are defined by the brainpoolP256r1 curve as specified in RFC 5639.

Any participant may generate a 264-bit compressed public ECDH key and publish the key on the Whiteflag network using a K(0)0A message. This allows any two participants, who have both published their public key, to generate the shared secret using their own private key and the other's public key.

If one of the participants publishes an K(2)0A message with updated key, the existing shared secrets with other participants expire and new shared secrets must be generated for and by each other participant.

Only one single participant's public ECDH key is considered valid at any point in time. Nevertheless, multiple K(0)0A messages may be sent to republish the public key. If a subsequent K(0)0A message contains a different public key, this must be interpreted as an K(2)0A message with an updated key (which should have been sent instead).

The shared secret may be used as a basis for encryption and authentication, whether for Whiteflag or not, but it should never be used directly as an encryption key.

5.2.3 Key and Token Derivation

Shared secrets (such as pre-shared, deterministically derived or ECDH generated secrets), must use the HKDF function defined in RFC 5869 to derive the actual tokens and encryption keys used for Whiteflag.

Note that it is crucial for security that shared secrets used as the input keying material for HKDF, have enough entropy, i.e. are sufficiently long (at least 16 bytes) and are practically indistinguishable from random data.

The HKDF function must use SHA-256 as the digest algorithm. Furthermore, the HKDF function takes three (3) parameters depending on the encryption method or token type: the key/token length, a usage specific salt and the the blockchain address as the info value.

- for authentication method 2 (token-based):
  1. the key length (token length) must be 32 bytes (256 bits)
  2. the salt value must be (hexadecimal): `420abc48f5d69328c457d61725d3fd7af2883cad8460976167e375b9f2c14081`
  3. the info value must be the binary blockchain address
- for encryption method 1 (`aes-256-ctr` with negotiated secret):
  1. the key length must be 32 bytes (256 bits)
  2. the salt value must be (hexadecimal): `8ddb03085a2c15e69c35c224bce2952dca7878770724741cbce5a135328be0c0`
  3. the info value must be the binary blockchain address
- for encryption method 2 (`aes-256-ctr` with pre-shared secret):
  1. the key length must be 32 bytes (256 bits)
  2. the salt value must be (hexadecimal): `c4d028bd45c876135e80ef7889835822a6f19a31835557d5854d1334e8497b56`
  3. the info value must be the binary blockchain address

Note that the salts are shown above in hexadecimal representation. Implementations must ensure that the data is correctly provided to the HKDF function, i.e. as binary information, not as a string. This is especially important for blockchains addresses, which appear in different encodings.

Using the 2.4.2.2 blockchain address `1C8KSK68SJjfDSBx9BpSx3qB3bePf23r77` as an example, this results in the following pseudocode for deriving authentication tokens:

```
tokenlength := 32
salt := 0x420abc48f5d69328c457d61725d3fd7af2883cad8460976167e375b9f2c14081
info := 0x007a0baf6f84f0fa7402ea972686e56d50b707c9b67b108866
secretToken := HKDF(sharedSecret, salt, info, tokenlength, digest="sha256")
```

The pseudocode for deriving encryption/decryption keys for the same blockchain address is as follows:

```
if (EncryptionIndicator == 1) then:
    keylength := 32
    salt := 0x8ddb03085a2c15e69c35c224bce2952dca7878770724741cbce5a135328be0c0
    info := 0x007a0baf6f84f0fa7402ea972686e56d50b707c9b67b108866
if (EncryptionIndicator == 2) then:
    keylength := 32
    salt := 0xc4d028bd45c876135e80ef7889835822a6f19a31835557d5854d1334e8497b56
    info := 0x007a0baf6f84f0fa7402ea972686e56d50b707c9b67b108866
key := HKDF(sharedSecret, salt, info, keylength, digest="sha256")
```

5.2.4 Message Encryption

Two encryption methods are currently defined by Whiteflag: the Advanced Encryption Standard (AES) using a 256-bit key in counter mode (CTR) with either an ECDH negotiated secret (method 1) or a pre-shared secret (method 2).

AES is a symmetric cipher that requires the same key for encryption and decryption. This key must be derived as described in the Key and Token Derivation paragraph.

5.2.4.1. Encryption Methods 1 and 2 Encryption methods 1 and 2 only differ in the key used for the encryption:

- Encryption method 1 uses the ECDH negotiated secret as described in the Key Agreement paragraph to derive the encryption key. This method can therefore only been used to send encrypted messages between the two blockchain accounts that have negotiated the shared secret.
- Encryption method 2 uses a pre-shared secret to derive the encryption key. It is outside the scope of this standard to define the key management and distribution of pre-shared keys. However, it is important for security that the pre-shared secret is sufficiently long and random.

Encryption of a Whiteflag message is performed with the following steps:

- derive the encryption key respectively from an ECDH negotiated secret or a pre-shared secret using the HKDF function as described in the the Key and Token Derivation paragraph;
- generate a unique 128-bit initialisation vector, e.g. with cryptographically secure random number generator;
- encrypt the compressed binary encoded message starting at bit 32 (the 33th bit) up to and including the last bit, using `aes-256-ctr` with the appropriate key and the just generated initialisation vector.

Below are the encryption steps in pseudocode, in which the key has already been derived as described in 5.2.3 Key and Token Derivation:

```
iv := generateRandomBits(128)
encryptedMessagePart := aes(encodedMessage[32...n], key, iv, mode="ctr")
encryptedMessage := encodedMessage[0...31] + encryptedMessagePart
```

The OpenSSL implementation of AES is the reference implementation for the usage of AES to encrypt and decrypt Whiteflag messages.

AES in counter mode requires a different initialisation vector for each message encryption, especially when reusing the same key. Therefore, a new random initialisation vector must be created for each message to be encrypted.

The encrypted message, with the correct encryption indicator set and the plaintext from the 5th byte onwards replaced with the ciphertext, must be sent as any other Whiteflag message.

The encrypted message must immediately be followed and referenced by a `K(3)11` or a `K(3)21` message containing the initialisation vector. Without the initialisation vector the message cannot be decrypted by the recipient. The `K` message must not be encrypted itself, and the duress indicator must be `0`, regardless of the duress indicator value of the referenced encrypted message.

5.3 Sending stand-alone Signs and Signals

To send a stand-alone message, Reference Code `0` must be used.

However, nothing prevents other senders to reference the message at any later point in time, thus creating a message sequence from a previously stand-alone message.

A message is sent by embedding it in a blockchain transaction. The way this is done is blockchain specific. Annex A contains some information about message embedding for a limited number of blockchains.

5.4 Referencing

5.4.1 Reference Options

5.4.1.1 Allowed Referencing between Message Types The table below shows which references types (as defined in Reference Indicator Field) may be used between the different message types, and which reference types are not allowed.

Referencing Message	P	E	D	S	I	Q	M
P	1/2/3/4/6/7/8/9	X	X	X	3/5	X	3/
E	X	1/2/3/4/6/7	3/5	5	X	X	5
D	5	X	1/2/3/4/6/7/9	X	5	X	5
S	3/5	X	X	1/2/4/6/7	3/5	X	X
I	X	X	X	X	1/2/3/4/6/7	X	X
Q	X	5	5	X	X	1/2/3/4/7/8/9	X
M	X	5	5	X	X	3	1/

Referencing Message	P	E	D	S	I	Q	M
F	3	3	3	3	3	3	3
R	3	3	3	3	3	3	3
A	X	X	X	X	X	X	X
K	3	3	3	3	3	3	3
T	T	T	T	T	T	T	T

X: prohibited *T*: a test message may make any reference for testing purposes

5.4.1.2 Allowed Referencing between Reference Types The table below shows which reference types (as defined in Reference Indicator Field) may be used to refer to an earlier message with a certain reference code to create a meaningful message sequence.

Referencing Message Code	0	1	2	3	4	5	6	7	8	9
0	X	X	X	X	X	X	X	X	X	X
1	Same	Same	Same	Same	Same	Same	Same	Same	Same	Same
2	Same	X	X	Same	X	Same	X	X	X	X
3	Same	X	Same	X	X	Same	X	X	X	X
4	Same	X	X	X	X	Same	Same	X	Same	Same
5	Both	X	X	X	X	Both	X	X	X	X
6	Other	X	Other	Other	Other	Other	X	X	X	X
7	Other	X	Other	X	X	Other	X	X	Other	Other
8	Other	X	Other	X	X	Other	X	X	X	X
9	Other	X	Other	X	X	Other	X	X	X	X

Same: reference code allowed by the *same* originator of the referenced message *Other*: reference code allowed by an *other* originator than of the referenced message *Both*: reference code is allowed by *both* the same or an other originator than of the referenced message *X*: prohibited

A message with Reference Code 0 (an original message) may not reference any message.

A message with Reference Code 1 (a recall message) may reference a message with any reference code of the same originator, meaning that a message with any reference code can be recalled.

A message with Reference Code 2 (an update message) may only reference a message with reference code 0, 3, or 5 meaning that only original, additional, and referring messages may be updated and only by the same originator.

A message with Reference Code 3 (an additional information message) may only reference a message with reference code 0, 2, or 5, for the same originator to provide additional information to original, updating and referring messages.

Reference Code 3 may be only used once to refer to the original message of the same message type. For example: S(0) < R(3) < R(3) and ... < [F(3)] < K11(3) << [F(3)] < K11(3) are allowed, but ... < R(3) < R(3) << R(3) is not. There is one exception: Reference Code 3 may be used to reference the same message type of types P, D, Q, and M more than once, when creating composite areas (see Composite Areas and Structures). In that case, the additional information must reference the initially referenced message and may not reference another message with additional information, e.g. D(0) < D(3) << D(3) <<< D(3)

A message with Reference Code 4 (a discontinue message) may only reference a message with reference code 0, 5, 6, 8, or 9 of the same originator, because an only originator may only discontinue an original, related, confirmation, comply or reject message. Discontinuing a message also implies that any later updates to that message are discontinued.

A message with Reference Code 5 (a referring message) may only reference a message with Reference Code 0 or 5, i.e. to refer to an original or another referencing message.

A message with Reference Code 6 (a confirmation message) may only reference a message with Reference Code 0, 2, 3, 4, or 5, i.e. confirming an original, updated, additional information, expiration, or related message.

A message with Reference Code 7 (an acknowledgement message) may only reference may only reference a message with reference code 0, 2, 5, 8 or 9, to acknowledge an original, updated, related, comply or reject message.

A message with Reference Code 8 (a comply message) may only reference a message with Reference Code 0, 2, or 5, i.e. to indicate the originator will act upon or in accordance with an original, updated, or a related message from

another originator.

A message with Reference Code 9 (a reject message) may only reference a message with Reference Code 0, 2, or 5, i.e. to indicate that the originator rejects, denies or does not agree with the referred original, updated, or a related message from another originator.

5.4.2 Message Sequences

When referencing messages are referenced themselves, message sequences may be created. The paragraphs below describe which message sequences, as allowed by the reference options described in Reference Indicator Field, provide meaningful information.

5.4.2.1 Authentication An A message may only reference another A message, in order to:

- recall the A(0) message with Reference Code 1;
- update the A(0) message with Reference Code 2, e.g. to provide new authentication information;
- provide additional information with Reference Code 3, when the authentication information is too long to provide in the VerificationData field of a single message;
- to discontinue the original A(0) with an A(4) message, meaning the originator leaves the network;
- to confirm the claimed identity of another originator by referencing its A(0) message with an A(6) message.

For example, an initial authentication of originator X using an URL of 30 bytes, which is confirmed by another party Y, later updated with an URL of the same length, and finally the account is closed, will result in the following message sequence:

```
A01(0)X < A01(3)X << A01(6)Y <<< A01(2)X < A01(3)X <<<<< A01(4)X
```

5.4.2.2 Updating Signs and Signals Conflicts and disasters are dynamic and events end or change. Therefore, the protocol allows to update signs and signals using message sequences with Reference Codes 2, 3 and 4.

The following example are illustrative for updating signs and signals and should be interpreted similarly when updating other signs and signals:

Message Sequence	Explanation
P10(0) < P10(2) << R01(3) <<< P10(4)	An original P10(0) message reports the presence of an ICRC-entity, which is subsequently updated with an P10(2) message, e.g. because the location changed. A reference is made to an ICRC-website about the entity to provide additional information using a R01(3) message. At a later point in time, the protective sign is withdrawn by using a P10(4) message to discontinue the original sign.

5.4.2.3 Relating Signs and Signals Events are often related. For example, a dangerous situation such as a wildfire may cause multiple buildings to be destroyed and multiple emergencies. The protocol allows to relate the different signs and signals for these events to create a better understanding of a situation.

A typical sequence of message will be a danger/disaster sign, followed by one or more status and emergency signals using Reference Code 5, which may be used to reference messages from any other account.

The following examples are illustrative for relating signs and signals and should be interpreted similarly when relating other signs and signals:

Message Sequence	Explanation
`D32(0)X < D43(5)X << E12(5)X << S26(5)Y <<< S25(5)Z`	An original `D32(0)` message reporting a rail accident is initially referenced two times: to indicate the accident coincides with a chemical explosion with a `D43(5)` message and to make an emergency call for fire & rescue assistance with a `E12(5)` message. Additional status messages relating to the accident or explosion may be sent, to report the status and locations of objects that are affected as a result of the accident or explosion.
`D15(0)X < E01(5)Y < E01(4)Y <<< S11(5)Y`	An original message `D15(0)` reporting a complex attack is referenced by somebody sending an `E01(5)` distress signal as a result of the attack. Later, when the person is in safety, the distress signal is discontinued with an `E01(4)` message and an `S11(5)` proof of life is sent.
`D52(0)X < R01(5)A << R01(5)B <<< R01(5)C`	An uncontrolled uprising is reported with a `D52(0)` message, and then referenced by three `R01(5)` messages from Other Orig.nators A, B and C to point to relating internet resources such as pictures of the uprising on social media.

5.4.2.4 Confirming Signs and Signals Conflict and disaster zones are chaotic and it is often difficult to assess a situation based on a single source of information. To improve the reliability of information, the protocol allows another originator to confirm of signs and signals posted on the network using Reference Code 6.

The following examples are illustrative for confirming signs and signals and should be interpreted similarly when confirming other signs and signals:

Message Sequence	Explanation
`D21(0)X < D21(6)Y << D21(6)Z`	An original message reporting a minefield is confirmed two times by different originators with two `D21(6)` messages referring to the original `D21(0)`.
`D21(0)X < D21(6)Y < D21(4)Y`	An expiration message `D21(4)` referring to a `D21(6)` confirmation message means that the original `D21(0)` message reporting a minefield cannot be confirmed anymore.

5.4.2.5 Composite Areas and Structures A composite area is a single area specified by multiple messages. A composite structure is a single structure specified by multiple messages. Composite areas and structures may be used, for example, when an area/structure is larger than the `ObjectSize` fields allows, or to create areas/structures with shapes other than the `ObjectType` field allows.

To report a new composite area or structure, the first message must have its Reference Indicator set to `0` (original message), and any following message in the sequence must reference the original message using Reference Indicator 3. The referencing messages must have the same Subject Code as the original message.

For example, a single mine field comprised of three area parts is reported on the network with the following message sequence:

`D21(0) < D21(3) << D21(3)`

Composite areas may only be created using messages with Object Code 20-29. Composite structures may only be created using messages with Object Code 30-39.

To reference a composite area or structure as a whole (e.g. to discontinue it using Reference Code 4 or to confirm it using Reference Code 6), the referencing message must reference the original message.

An update message (Reference Code 2) may reference individual messages of the sequence, and thus updating each part of the composite area/structure separately. A recall message (Reference Code 1) may either reference the original message, or individual messages of the sequence.

5.4.2.6 Acknowledging messages Acknowledgements may be sent to inform others of having 'received' a certain message, i.e. to acknowledge that a certain message has been witnessed, e.g. to notify the receipt of updated information or a rejection. Acknowledgements may refer to an original, updated, related, comply or reject message and can be used in message sequences.

Message Sequence	Explanation
`D21(0)X < D21(2)X < D21(7)Y`	An acknowledging message `D21(7)` referring to a `D21(2)` update message, means that the update to `D21(0)` message has been acknowledged.
`Q20(0)X < Q20(9)Y < Q20(7)X`	An Area Access request signal 'Q20(0)' is rejected by a Q20(9) message, which is then acknowledged by an Q20(7) message

5.4.2.7 Complying to messages The 'comply' message reference code provides the option to an originator to indicate they will act upon or in accordance with an original, updated, or a related message from another originator. This referencing option may be useful to respond to other originators' message to inform or notify them about an action to be undertaken.

Message Sequence	Explanation
`E11(0)X < E11(8)Y`	An Emergency Signal `E11(0)` indicating that urgent medical assistance is required, is responded to by somebody complying with the ask for assistance.
`P02(0)X < P02(8)Y << P02(8)Z`	A request for negotiation `P02(0)` will be complied to by two different originators respectively sending a `P02(8)` message.

5.4.2.8 Rejecting messages The 'reject' message reference code may be used to indicate that the originator rejects, denies or does not agree with the referred original, updated, or a related message from another originator.

Message Sequence	Explanation
`Q20(0)X < Q20(9)Y`	An Area Access request signal 'Q20(0)' is rejected by a Q20(9) message
`D21(0)X < D21(9)Y`	An original message reporting a minefield is rejected by a different originators with a `D21(9)` messages referring to the original `D21(0)`.

5.5 Testing

Test messages must be disregarded for any other purposes than testing. Test messages should only be used on testing networks of blockchains.

A test message may reference any other message for testing purposes.

Annex A. Blockchain layer specific details

This Annex is meant to provide additional information and considerations about the use of blockchain networks for Whiteflag Protocol end-users and for creating implementations.

A.1 Implementation Guidelines

A.1.1 Completeness of the blockchain database

When using the Whiteflag Network as a source for improving situational awareness, it is important for end-users to have a complete view on all messages that have been sent in a given context. It is advised for end-users to assess if all (relevant) blocks have been processed and no message has been omitted or skipped, as this might result in inaccurate interpretation of information.

A.1.2 Data encapsulation in transactions

The Whiteflag Network is created as a blockchain overlay network by Whiteflag applications embedding Whiteflag messages in blockchain transactions. The underlying blockchain network(s) therefore need(s) to allow for the encapsulating of data in a transaction.

For security reasons, it is currently advised to implement the Whiteflag Protocol only on blockchain networks that use a proof-of-work consensus model.

There exist multiple blockchain networks that provide this functionality. See the table below for several examples:

Blockchain	Message embedding	Maximum message length	Signature algorithm	Transaction hash
Bitcoin	`OP_RETURN`	80 bytes	ECDSA secp256k1	256 bits (64 hexadecimals)
Ethereum	`data` field	(dynamic)	ECDSA secp256k1	256 bits (64 hexadecimals)

In the case of Bitcoin, the `OP_RETURN` field of a raw transaction allows for the encapsulation of data constrained to a maximum 80 bytes.

In the case of Ethereum, the `data` field of a raw transaction allows for the encapsulation of a dynamic amount of data that is relative to the `gasLimit` of individual blocks. In practice, this field can generally store multiple kilobytes of data.

Message formatting within the Whiteflag Protocol has been structured to not require more than 80 bytes for sending signs and signals. Reference messages and/or concatenation may be used in cases where providing additional data is required.

Please note that the above parameters may change over time as the codebases of blockchain networks evolve, in which case implementations of the Whiteflag Protocol may need to be updated accordingly.

A.1.3. Monitoring of Dynamic Blockchain Parameters

Blockchain networks are highly dynamic distributed computing networks that, because of their decentralized architecture, rely both on actively and continuously maintained consensus by its participating validating nodes, as well as on the peer-to-peer infrastructure for transceiving data over the network.

It is advisable to monitor several parameters when implementing the Whiteflag Protocol to ensure the underlying blockchain network(s) is/are operating correctly.

A.1.3.1 Relaying and Processing of Messages The following parameters concern the relaying and processing of Whiteflag messages by the underlying blockchain network(s).

Peer-to-Peer Connectivity Monitoring of the amount of peers a message sending Whiteflag account connects to, to ensure messages are relayed correctly. Many blockchain networks provide a list of nodes that have proven long-term reliability. It is up to the Whiteflag participant to decide how many and which peers to connect to.

Transaction Costs Each message will inflict a minimal Transaction Cost to be paid for the network to process the transaction. These costs usually consist of a minimum transaction amount and a transaction fee. End-users need to ensure the proper amount of funds are included for each message that is sent.

Block Time Each blockchain networks' consensus protocol defines an average Block Time the network targets for creating new blocks. Monitoring of block creation time can assist in determining the expected validation time of a message.

Transaction Relay Time Monitoring of the number of transactions on the blockchain network pending verification. In cases of very high transactional load, transactions may require longer processing times or increased fees.

Number of Block Confirmations Monitoring the number of blocks that have been added to the shared database *after* the block containing a Whiteflag message. It is up to the Whiteflag participant to determine how many blocks *deep* a message needs to be for it to be considered secured in the blockchain.

Blockchain Protocol Changes Blockchain networks rely on their respective protocols for verifying transactions on the network. In case of a change in protocol, features may change causing Whiteflag Protocol implementations to require updating.

A.1.3.2 Blockchain Security The following parameters concern the security of the underlying blockchain network(s):

Network hash-rate Monitoring of the aggregated computational power of the number of transaction validating nodes.

Changes in Network Hash-Rate Monitoring whether there are significant changes in the network's hash-rate and if so, take appropriate measures.

Hash-Rate Distribution Monitoring the distribution of the total hash-rate amongst the participating validating nodes. In case of a validating node (or a collection of nodes) representing 51% of the total network hash-rate, take appropriate measures.

Consensus Protocol Changes Blockchain networks rely on their respective consensus protocols for the proper validation of transactions. Major changes to a consensus protocol may have significant impact to a blockchain network as a whole.

Chain Reorganisation Dynamics Malicious actors may perform attacks on a blockchain network by causing rapid and/or large-scale chain reorganisations. Many blockchain network prescribe a chain "re-org limit" and have implemented other measures to mitigate such events. It is advised to assess resiliency to such attacks per individual blockchain network.

A.2. General considerations related to blockchain network functionality

A.2.1 Consensus protocols

Blockchain networks using the proof-of-work consensus model (Nakamoto consensus) timestamp transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The timestamp proves that the data must have existed at the time in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp

reinforcing the ones before it. The hash chain of blocks provides an unchangeable historical record, which is verifiable by anyone.

Blockchain networks using different consensus protocols are currently in an experimental phase and have not yet been proven to work securely at large scale. At this point, it is therefore not advised to use an other type of blockchain network than those using a proof-of-work consensus protocol.

A.2.2 Market dynamics

Blockchain networks require a native currency (often called crypto-currency) to incentivize validating nodes to expend resources on doing the proof-of-work. Transactions containing Whiteflag messages are sent by spending only the minimum amount of native currency required by the blockchain network's protocol to effect a valid transaction, plus an additional fee.

While these amounts may be relatively small, the market dynamics related to these native currencies can cause price volatility, bear/bull dynamics or even trade-halts, which may cause effect usability.

A.2.3 Private Blockchain Networks and Permissioned Distributed Ledgers

Please note that the Whiteflag Protocol is specifically designed to be implemented on open blockchain networks. Private blockchain networks, or permissioned distributed ledgers, do not adhere to the design principles of the Whiteflag Network, as these systems require a trusted party (i.e. trusted nodes) to establish consensus within the network and do not provide open access, and should therefore not be used.

A.2.4 Anonymous Transactions

Blockchain networks using digital signature schemes that enable fully anonymous transactions could theoretically be used to anonymously send messages on the Whiteflag Network. Examples of such blockchains employ techniques like zero-knowledge-proofs and ring signatures to create conditions of untraceability and unlinkability. These type of transactions do not adhere to the principles of this standard, as they limit message authentication, and should therefore not be used.

Annex B. JSON Schema of Whiteflag Messages

(*contained in separate file*)

Annex C. Example JSON Authentication Objects

The JSON representation of an A1 message sent using an example static Base58 blockchain address 1C8KSK68SJjfDSBx9BpSx3qB3bePf23r77 may be as follows:


```
{
    "MetaHeader": {
        "example": true
    },
    "MessageHeader": {
        "Prefix": "WF",
        "Version": "1",
        "EncryptionIndicator": "0",
        "DuressIndicator": false,
        "MessageCode": "A",
        "ReferenceIndicator": "0",
        "ReferencedMessage": "0000000000000000000000000000000000000000000000000000000000000000"
    },
    "MessageBody": {
        "VerificationMethod": "1",
        "VerificationData": "https://organisation.int/whiteflag"
    }
}
```


The related JSON formatted object with the authentication information to be signed and published at the provided URL https://organisation.int/whiteflag would be:


```
{
    "addr": "1C8KSK68SJjfDSBx9BpSx3qB3bePf23r77",
    "orgname": "Organisation Name",
    "url": "https://organisation.int/whiteflag"
}
```


This object is then used as the payload to create a JSON Web Signature (JWS) object that includes a digital signature created with the private key of the Bitcoin address. The compact serialization specified by the JWS standard would normally result in the following token:


```
eyJhbGciOiJFUzI1NiJ9.eeyJhZGRyIjoiMUM4S1NLNjhTSmpmRFNCeDlCcFN4M3FCM2JlUGYyM3I3N
yIsIm9yZ25hbWUiOiJPcmdhbmlzYXRpb24gTmFtZSIsInVybCI6Imh0dHBzOi8vb3JnYW5pc2F0aW9u
LmludC93aGl0ZWZsYWcifQo.XWBRA1TrCxs8tpep1lLPcmpp9JlO_AOTJB5ULOROvadje3SgAsfkFEj
E2DoHGpWJ_zNGlEPBtdUQo9MEypIp2Q
```


However, for practical purposes the object published at the provided URL must be a flattened JWS JSON Serialization object:


```
{
    "protected": "eyJhbGciOiJFUzI1NiJ9",
    "payload": "eeyJhZGRyIjoiMUM4S1NLNjhTSmpmRFNCeDlCcFN4M3FCM2JlUGYyM3I3NyIsIm
                9yZ25hbWUiOiJPcmdhbmlzYXRpb24gTmFtZSIsInVybCI6Imh0dHBzOi8vb3JnY
                W5pc2F0aW9uLmludC93aGl0ZWZsYWcifQo",
    "signature": "XWBRA1TrCxs8tpep1lLPcmpp9JlO_AOTJB5ULOROvadje3SgAsfkFEjE2DoHG
                  pWJ_zNGlEPBtdUQo9MEypIp2Q"
}
```


The equivalent of a fully unserialized JSON object representation is provided here as human readable example:


```
{
    "protected": {
        "alg": "ES256"
    },
    "payload": {
        "addr": "1C8KSK68SJjfDSBx9BpSx3qB3bePf23r77",
        "orgname": "Organisation Name",
        "url": "https://organisation.int/whiteflag"
    },
```

```
    "signature": "XWBRA1TrCxs8tpep1lLPcmpp9JlO_AOTJB5ULOROvadje3SgAsfkFEjE2DoHG
                  pWJ_zNGlEPBtdUQo9MEypIp2Q"
}
```


Note that the used algorithm (ES256, i.e. ECDSA using P-256 and SHA-256, in line with the underlying blockchain) is included in the protected header.

Annex D. Definitions

Below is an overview of terms as used in this standard within the context of Whiteflag.

Term	Description
Account	The identifier under which an *originator* utilises one or more blockchain *addresses*.
Address	The identifier that corresponds to a cryptographic key pair to perform *transactions*, i.e. to send *messages*, on a *blockchain*.
Application	Any software, system or device utilising (part of) the Whiteflag Protocol.
Authentication	The confirmation that a claimed characteristic of an entity is actually correct, in this case used to verify the *originator* of a *message*.
Blockchain	A distributed and continuously growing list of grouped *transactions*, called blocks, which are linked and secured using cryptography.
Confidentiality	The assurance that the information contained in a *message* is not made available or disclosed to entities as required by the originator.
Data Integrity	Verification that the information contained in a *message* is received completely and unaltered.
Disaster	A sudden, calamitous event that seriously disrupts the functioning of a community or society and causes human, material, and economic or environmental losses that exceed the community's or society's ability to cope using its own resources.
Message	An atomic piece of Whiteflag data as defined in this standard and incorporated in a single *transaction* on a *blockchain*.
Network	The usage of the Whiteflag protocol on a specific *blockchain*, or all Whiteflag *participants* on a specific *blockchain*.
Non-repudiation	The undeniable proof that a *message* has been sent by an *address*.
Open Standard	A standard that is public and without any restriction in their access or implementation.
Originator	A *participant* that is the sender of a *message*.
Participant	A person or an organisation using the Whiteflag protocol; a participant may have multiple *blockchain accounts*.
Protective Sign	A symbol used to mark persons and objects under the protection of treaties international law and regulations.
Security	The measures to protect and preserve the *integrity* and *availability* of the messages, and, if required by the *originator*, the *confidentiality*.
Transaction	A record on a *blockchain* (containing a single Whiteflag *message*).

Annex E. Signs & Signals Equivalents

(*contained in separate repository*)

Annex F. Example Use Cases

Use case 1: NGO Area Access to Provide Aid

A humanitarian convoy is carrying emergency assistance for affected populations in a conflict zone. First of all, the authenticity of the blockchain account of the aid organisation is validated with an A2(0) message and can be reinforced on a regular basis with A2(2) updates.

In order to safely reach their destination, the aid organisation needs to request area access in advance to all parties to the conflict. These requests are made using Q messages, with which the following information is provided: trajectory, dates, vehicles and personnel. The aid organisation is able to send these request encrypted to each party individually by generating a unique encryption key for that party using the K0A messages on the blockchain.

Parties can indicate the receipt and compliance with the request by using reference codes 7 and 8. Last but not least, a party may withdraw its compliance with the request at any time by sending a Q(9) deny message. This would result in the following sequence:

```
Q20(0)A < Q20(3)A ... << Q(7) <<< Q(8) ... <<<< Q(9)
```

While on the road, the convoy is able to quickly inform other humanitarian actors of potential changes in the environment as well as receive as much as possible live information while driving, by using protection and danger signs, e.g. the P11 protective sign, D13 bomb attack, D21 unexploded ordnance, DA5 bomb attacks, D51 uncontrolled uprising, etc.

Once on location, a protective sign is sent. In order to achieve meaningful humanitarian action, the NGO also communicates the type of assistance provided using M messages to avoid duplication and inefficiency of humanitarian aid:

```
P11(5) < M(5)
```

Use case 2: Protected Sites and Critical Infrastructures

Since messages are persistent on a blockchain, an organisation can utilize the Whiteflag Protocol to create a public digital register of sites that are protected under international law using P messages. In addition, I messages are used for notifications about infrastructures that are not necessarily protected under international law, but are important for NGO operations or for affected populations and refugees: e.g. water treatment facilities, power plants, food distribution location, shelter, NGO warehouses, using I messages.

This information might be shared to prevent unintended air attacks, but also to share information between NGOs and the UN. For example, an NGO maintains a continuously updated overview of critical infrastructures and a register of hospitals on the Whiteflag network, including status updates. It is an operational decision to disclose or encrypt the information to prevent abuse.

Currently, this type of information is not always confirmed or denied afterwards. Using Whiteflag, other parties can confirm the information by using reference code 6, e.g.:

```
I(0) < I(6) ... P31(0) < P31(6)
```

A message sequence where a hospital is identified, a party indicates it will comply with its protected status, and the status of the hospital is periodically updated, will look like:

```
P31(0) < P31(8) << S(5) ... < S(2)
```

Another example is a local cultural institute C that can create a public digital register of monuments in its city or village by posting a protective sign at each monuments location, using P52 messages (protective sign for a cultural property).

When a protected site or critical infrastructure ever comes under attack, this can be broadcasted immediately by anyone using the Whiteflag Protocol, sending for example a D11 message (structure under ground attack), resulting in the following message sequence:

```
P52(0)C < D11(5)
```

Were the attack undertaken by a party that intends to comply with international law (for which reason such a party might very well actively use the Whiteflag Protocol), that party would be immediately informed of their mistake allowing them to cease fire before any more damage is done.

If the attack is performed by a non-compliant armed group, the Whiteflag Protocol ensures that the attack is recorded and known to the international community, allowing for example peacekeeping forces to react and/or providing evidence for the war crime.

Use case 3: Peacekeeping Forces Confirm a Mine Field Reported by Locals

The Whiteflag Protocol allows to digitally mark dangerous zones, such as minefields or areas where unexploded ammunition is present. Suppose local civilians are aware of an area where mines are present. A local aid worker A might report such an area by sending one or more `D21` messages using a special app on her smartphone. The information about the dangerous area is available immediately to civilians, NGOs, journalists etc., even though the information about the mine field might not yet considered accurate or true by everybody.

Based on the original message, a UN peacekeeping unit or UNMAS (U) plan to investigate the area. Once they can confirm it is a minefield, a higher-level UN headquarters sends a confirmation message using a UN information system with a Whiteflag interface through a well-known and validated UN blockchain account.

A link to an UNMAS website with additional information (mine types, quantity, level of danger, stay clear area, current de-mining status) is provided using an `R` message, which can be resent when the web page is updated.

This scenario resulted in the following message sequence:

`D21(0)A < D21(6)U < R(3)U ... < R(2)U`

It would be very easy for anyone to develop a web application to actively monitor all these sorts of messages and project this information on a map, creating broad awareness of the danger among aid workers and local populations.

Use case 4: Proof of Life by a Journalist

A journalist working in a combat zone wants to let his news agency know where he is on a regular basis. To do this, the journalist makes his blockchain account known by sending an `A2` message containing a token that was provided to him by his news agency. He can now be positively identified on the network by his news agency, while his identity is not made public.

From now on, the journalist can send an `S11` (proof-of-life status message) on a regular basis, either with or without his location. For extra security, it is possible to encrypt the message to ensure only recipients know where he is.

Use case 5: Negotiate a Cease Fire between Fighting Parties

Warring parties can use Whiteflag to negotiate a cease fire, e.g. to be able to provide medical care in a specific area for a certain amount of time. In the following example sequence, party A requests a cease fire using a `Q` message with additional information included in an `F` message; parties B and C agree with the request by responding with reference code 8:

`Q(0)A < F(3)A << Q(8)B <<< Q(8)C`

Note that Whiteflag is limited to fixed messages and in itself not suitable to conduct detailed negotiations. However, even though other communication channels are used to come to a cease fire, Whiteflag allows to undeniably record the request and any outcomes.

Use case 6: Armed Group under Air Attack wants to Surrender

Imagine an armed group A is under a heavy air attack, and the remaining combatants want to surrender. However, it is very unlikely that they are in radio contact with the pilots, and the pilots might not see a physical white flag.

The Whiteflag Protocol can be used to send a message to the attacking party, without knowing any contact details. An `A2(0)` message is sent, or has been sent before, referring to some sort of social media channel that is known to be used by the armed group. The blockchain account is now linked to this group, and a `P03(0)` surrender message is sent.

If a headquarters or command centre of the attacking party M is listening on the Whiteflag Network for situational awareness, they will be instantaneously informed of the digital white flag on the location under attack. Of course, the trustworthiness of the message needs to be evaluated based on earlier communications from the same account, other

intelligence, confirmation by other parties on the network, the actual behaviour of the armed group at the location, etc.

Either way, the information can be relayed over a military data link to the pilots in the aircraft, who might then be able to visually confirm a white flag. Once that is the case, the attacking party M can send both a confirmation and an acknowledgement. This scenario resulted in the following message sequence:

```
A2(0)A < P03(0)A < P03(6)M << P03(7)M
```