The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

ROBUST COMPRESSION OF FIXED-LENGTH HOUSEKEEPING DATA

DRAFT RECOMMENDED STANDARD

CCSDS 124.0-R-1

RED BOOK

June 2022

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

ROBUST COMPRESSION OF FIXED-LENGTH HOUSEKEEPING DATA

DRAFT RECOMMENDED STANDARD

CCSDS 124.0-R-1

RED BOOK

June 2022

AUTHORITY

Issue:	Red Book, Issue 1
Date:	June 2022
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the email address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
Email: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

This Recommended Standard specifies a robust method for lossless compression of fixed length housekeeping data and a format for storing the compressed data.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be sent to the CCSDS Secretariat at the email address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (BELSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- China Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Netherlands Space Office (NSO)/The Netherlands.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 124.0-R-1	Robust Compression of Fixed-Length Housekeeping Data, Draft Recommended Standard, Issue 1	June 2022	Current draft

CONTENTS

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to establish a Recommended Standard for a low complexity, robust data compression algorithm applied to fixed-length binary vectors (as commonly arise in telemetry housekeeping data) and to specify the compressed data format.

Data compression is used to reduce the volume of digital data to achieve benefits in areas including, but not limited to,

a) reduction of transmission channel bandwidth;

b) reduction of the buffering and storage requirement;

c) reduction of data-transmission time at a given rate.

In the particular case of housekeeping telemetry data, compression can be used to increase the net retrieval of housekeeping information within the available bandwidth.

1.2 SCOPE

The characteristics of the data compression algorithm are specified only to the extent necessary to ensure multi-mission support capabilities. The specification does not attempt to quantify the relative bandwidth reduction, the merits of the approaches discussed, or the design requirements for encoders and associated decoders.

This Recommended Standard addresses lossless data compression of fixed-length binary vectors. The compression method specified herein is effective when a subset of the bits tend to be unchanged from one vector to the next. Housekeeping telemetry and some payload data regularly exhibit this property. The algorithm encodes fixed-length input binary vectors losslessly, and has no pre-conditions on the format of the input data or the number of binary vectors required to be processed before compressed output data can be produced. Consequently, the algorithm can be used while compressing files, packet stores, or real-time data streams.

Since some packet losses are normal for operations relying on housekeeping telemetry, the Recommended Standard includes mechanisms to continue the decompression of subsequent packets following the loss of one or more packets. Most often, decompression can continue as soon as the next packet is received. When too many packets are lost sequentially, synchronization between the encoder and the decoder may be temporarily broken. Decompression may then resume normally only after a packet with sufficient information is received. The Recommended Standard also guarantees that the sequence of packets successfully received between synchronization and the last lost packet can be losslessly decoded.

1.3 APPLICABILITY

This Recommended Standard applies to data compression applications of space missions anticipating packetized telemetry cross support. In addition, it serves as a guideline for the development of compatible CCSDS Agency standards in this field, based on good engineering practice.

1.4 RATIONALE

The number of telemetry parameters in spacecraft housekeeping telemetry is constantly increasing. The bandwidth available to download that information is not increasing at the same rate. Operators must therefore make increasingly hard choices about which parameters to downlink in which situations and at which sampling rates. Compressing the information would help, but most missions download housekeeping in a 'fire and forget' mode of operation; that is, occasional packet losses are expected. This makes traditional compression schemes such as zip or tar unsuitable. Also, any software-based compression algorithm would usually be executed on flight processors, which are typically not very powerful and which run flight-critical real-time applications in parallel. Finally, housekeeping data is transmitted to the ground in real time but also stored on-board and then dumped during visibility. These two data streams compete for the same bandwidth; therefore compressing both with the same algorithm would be ideal. The rationale for this standard is therefore to provide effective housekeeping-data compression that can work on both the stored and real-time data streams, is robust to occasional packet loss, and has a low CPU usage.

The concept is to compress each original housekeeping packet into a smaller packet when it is generated. The compressed packet can be stored or transmitted in real time, as normal. The algorithm must therefore be fast enough to compress a typical stream of housekeeping packets being generated one by one. The design ensures speed by making sure it can be implemented using very-low-level processor instructions (XOR, OR, AND, etc.). Robustness with respect to packet loss is built into the concept, as is a mechanism to adapt the amount of robustness required so it can be altered according to the situation.

The concept and rationale for the robust compression algorithm described herein are described in further detail in reference [C6].

1.5 DOCUMENT STRUCTURE

This document is organized as follows:

a) Section 1 provides the purpose, scope, applicability, and rationale of this Recommended Standard and identifies the conventions and references used throughout the document. This section also describes how this document is organized. A brief description is provided for each section and annex so that the reader will have an idea of where information can be found in the document.

b) Section 2 provides an overview of the compressor.

c) Section 3 specifies the input for the compressor and the parameters used to control it.

d) Section 4 specifies the mask update stage of the compressor.

e) Section 5 specifies the encoding stage of the compressor and the format of the compressed data.

1.6 CONVENTIONS AND DEFINITIONS

1.6.1 MATHEMATICAL NOTATION AND DEFINITIONS

For the purposes of this Recommended Standard, the following definitions and conventions apply. Many other terms that pertain to specific items are defined in the appropriate sections.

An integer is denoted by an italic upper- or lower-case letter without accent, for example, a.

A single bit is denoted by a letter with a single dot accent, for example, $\dot{a}$. Boldface type is used to indicate a binary vector, that is, a sequence of bits. An uppercase boldface letter identifies a binary vector of fixed length, for example, **B**; a lower-case boldface letter identifies a binary vector of variable length, for example, **b**. A binary vector consisting of all '0' bits is denoted by a boldface 0, that is, **0**. In this document, all fixed length binary vectors have the same length. A binary vector of length zero is denoted by $\emptyset$**.** Binary vectors that have fixed bit values are denoted with single quotes, for example, '10010'.

When relevant, a time index t is included as subscript. For example, $\boldsymbol{C}_t$ indicates a fixed-length binary vector that is computed on time index t. The time index refers to the sequence number of the input binary vector rather than an absolute time.

A subscript is also used to identify a bit within a binary vector. For example, $\dot{a}_i$ indicates the bit in position i of binary vector $\boldsymbol{a}$. It is clear from context whether a subscript refers to a time index or a bit position within a binary vector.

In this document, the following convention is used to identify each bit in an N-binary vector. The first bit in the vector to be transmitted (i.e., the most left justified when drawing a figure) is defined to be 'bit N-1', the following bit is defined to be 'bit N-2', and so on down to 'bit 0'. When the binary vector is used to express an unsigned binary value (such as a counter), the Most Significant Bit (MSB), bit N-1, shall correspond to the highest power of two, that is, 2^{N-1}. Similarly, the Least Significant Bit (LSB), bit 0, shall correspond to the lowest power of two, that is, 2^0.

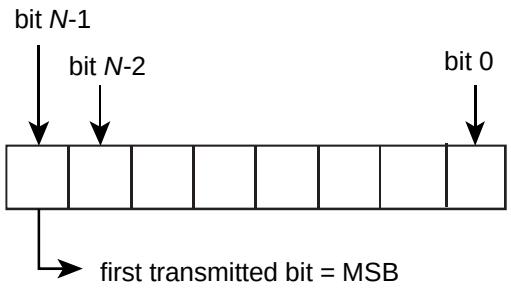

Figure 1-1: Bit Numbering Convention

Given a length-N binary vector $\boldsymbol{a}$, the notation $\boldsymbol{a}^{\ll}$ denotes the left bit-shift of $\boldsymbol{a}$, that is,

$$\boldsymbol{a}^{\ll} = \{\dot{a}_{N-2}, \dot{a}_{N-3}, \ldots, \dot{a}_1, \dot{a}_0, 0\}, \tag{1}$$

$\sim\boldsymbol{a}$ denotes the bit-wise inverse of $\boldsymbol{a}$, and $< \boldsymbol{a} >$ denotes the vector formed by reversing the order of the bits in $\boldsymbol{a}$. For example, if $\boldsymbol{a}$ = '10111', then $\boldsymbol{a}^{\ll}$ = '01110', $\sim\boldsymbol{a}$ = '01000', and $< \boldsymbol{a} >$ = '11101'.

$\mathrm{H}(\boldsymbol{a})$ denotes the Hamming weight of binary vector $\boldsymbol{a}$. That is, $\mathrm{H}(\boldsymbol{a})$ is the number of '1' bits in $\boldsymbol{a}$.

The K-bit unsigned binary representation of nonnegative integer $X \leq 2^K - 1$ is denoted $\mathrm{BIT}_K(X)$. For example, $\mathrm{BIT}_5(3)$ = '00011'.

Given two length-N binary vectors $\boldsymbol{a}$, $\boldsymbol{b}$, the logical disjunction of $\boldsymbol{a}$ and $\boldsymbol{b}$, denoted $\boldsymbol{a}$ OR $\boldsymbol{b}$, is a vector $\boldsymbol{c}$, where

$$\dot{c}_i = \begin{cases} 0, & \text{if } \dot{a}_i = \dot{b}_i = 0 \\ 1, & \text{otherwise} \end{cases} \quad i = 0, 1, \ldots, N-1, \tag{2}$$

the exclusive or of $\boldsymbol{a}$ and $\boldsymbol{b}$**,** denoted $\boldsymbol{a}$ XOR $\boldsymbol{b}$, is a vector $\boldsymbol{c}$, where

$$\dot{c}_i = \begin{cases} 0, & \text{if } \dot{a}_i = \dot{b}_i \\ 1, & \text{if } \dot{a}_i \neq \dot{b}_i \end{cases} \quad i = 0, 1, \ldots, N-1, \tag{3}$$

and the logical conjunction of $\boldsymbol{a}$ and $\boldsymbol{b}$**,** denoted $\boldsymbol{a}$ AND $\boldsymbol{b}$, is a vector $\boldsymbol{c}$, where

$$\dot{c}_i = \begin{cases} 1, & \text{if } \dot{a}_i = \dot{b}_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad i = 0, 1, \ldots, N-1. \tag{4}$$

The concatenation of binary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted $\boldsymbol{a} \parallel \boldsymbol{b}$.

For any real number β, the largest integer N such that $N \leq \beta$ is denoted by

$$N = \lfloor \beta \rfloor. \tag{5}$$

1.6.2 NOMENCLATURE

1.6.2.1 Normative Text

The following conventions apply for the normative specifications in this Recommended Standard:

a) the words 'shall' and 'must' imply a binding and verifiable specification;

b) the word 'should' implies an optional, but desirable, specification;

c) the word 'may' implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

1.6.2.2 Informative Text

In the normative sections of this document (sections 3-5), informative text is set off from the normative specifications either in notes or under the ‘Overview’ subsection headings.

1.7 REFERENCES

This document contains no normative references.

2 OVERVIEW

2.1 GENERAL

This Recommended Standard specifies a method for losslessly compressing a sequence of fixed-length input binary vectors into a sequence of variable-length output binary vectors. The method provides effective compression when a subset of the bits in the input vectors tend to be unchanged from one input vector to the next, as is often the case for housekeeping telemetry and some spacecraft payload data. The algorithm imposes no requirements on the format of the input data except that the input vectors must be fixed length. The compressor has no latency – that is, an encoded binary output vector is produced for each input vector. Consequently, the compressor can be applied to the live transmission of a stream of binary vectors. With live stream transmission, low power, and memory requirements in mind, the compressor has been designed so that the mathematical operations required can be implemented with very low-level bitwise operations such as ORs, XORs, ANDs, and bit shifts.

For each input bit vector, the compressor updates a state variable binary vector, having the same length as the input binary vector, called the *mask*. Each bit in the mask indicates whether the corresponding bit in the input binary vector is classified as 'predictable' or 'unpredictable'. When a bit at a given position in an input vector differs from the corresponding bit of the previous input vector, then that position is classified as unpredictable. Unpredictable positions cannot be reclassified as predictable until a user defined parameter called *new mask flag* is set to one. Therefore the decompressor can assume that all bits in positions classified as predictable have the same value as in the previous input binary vector, while unpredictable bits may not.

Given the input vector length, the decompressor requires three additional elements of information to reconstruct a given input binary vector:

a) the last successfully reconstructed binary vector in the series;

b) a mask synchronized with the one used to compress the input binary vector;

c) the unpredictable bit values associated with the input binary vector.

The compressor always includes mask change information in the output binary vector from which the decompressor can keep its own mask synchronized. This can work both forwards and backwards in a time series of output binary vectors. The compressor can also opt to encode the entire mask as part of an output binary vector. The compressor includes the unpredictable bit values associated with the input binary vector in the output binary vector or it encodes the original input binary vector. With this information, the decompressor can synchronize its mask so that the unpredictable bit values can be allocated to their correct positions. In this way, the last successfully reconstructed binary vector can be updated to reconstruct the associated input binary vector.

The algorithm runs in discrete cycles that are triggered by the arrival of a new input binary vector, and each of which consists of a mask update step followed by an encoding step. In the

mask update step, any bit position in which a bit value changes with respect to its previous value is classified as unpredictable. In parallel, a new mask, called *build*, is constructed following the same rules as for the mask. When the user-defined input parameter new mask flag is set to one, it results in the present mask being replaced by build. Build is then reset to declare all bit positions as predictable. In this way, it is possible for bit positions to make the transition from predictable to unpredictable every cycle, and from unpredictable to predictable only on the cycle when the new mask is requested. This scheme allows the system to track behavior changes of individual bit positions over time.

In the encoding stage, a variable-length output binary vector is produced by concatenating three binary vectors. The first vector encodes the positions in the mask vector that have changed over a number of cycles and indicates whether each change resulted in a predictable or an unpredictable classification. The second vector encodes the entire mask, depending on the user-defined parameter *send mask flag*. The third vector either encodes the bit values of the unpredictable bits or includes a copy of the input binary vector, depending on the user-defined parameter *uncompressed flag*.

The number of cycles that are included in the mask change information, called the *effective robustness level*, is encoded in the output binary vector. This describes the robustness to data loss guaranteed for each output vector: the mask can be synchronized even if the number of consecutive output binary vectors lost immediately before this output bit vector is equal to, or less than, the effective robustness level. The user can specify a minimum value for the effective robustness level by setting the user-defined parameter *minimum required effective robustness level*.

It should be noted that the decompressor is not required to actively change user defined parameters as all the information required for decompression is contained in the output bit vectors. This allows the compressor to change them periodically or on some other condition.

An overview of the whole compression process is given in figure 2-1.

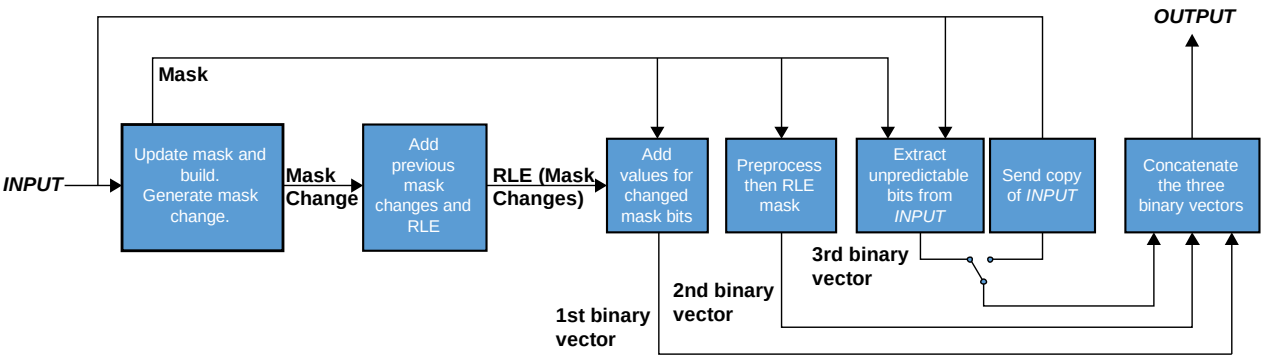

Figure 2-1: Overview of Compression Process

2.2 DATA TRANSMISSION

The Recommended Standard describes how to compress a single binary vector stream. If more than one bit stream is compressed, the identification of these streams is mission specific. For example, compressed data may be transmitted in space packets (reference [C1]) and application process identifiers (reference [C1]) could be used to identify different streams.

The Recommended Standard describes a mechanism to ensure that the compressor and decompressor remain synchronized in the event of the loss of a configurable number of sequential output binary vectors. However, it does not provide a mechanism for identifying the number of sequential output binary vectors that were lost. Such mechanisms are assumed to be mission specific. As in the prior example, if space packets are used, then Packet Sequence Counters (reference [C1]) could be used to support this function.

The effects of a loss of synchronization between compressor and decompressor, or data corruption, can result in the decompressor not knowing where the header of the following output binary vector is located. This Recommended Standard does not incorporate sync markers or other mechanisms to flag the header of the next output binary vector in those cases. Such mechanisms are mission specific. Space Packets with packet length fields (reference [C1]) could be used, for example.

The effects of a small error can result in the loss of synchronization between compressor and decompressor. Although the Recommended Standard provides mechanisms to resynchronize them, such an event will introduce a delay into the decompression, and some output binary vectors may not be able to be decoded. Therefore measures should be taken to minimize errors during the transmission of the output binary vectors.

In case the encoded bitstream is to be transmitted over a CCSDS space link, several protocols can be used to transfer an output binary vector, including, but not limited to:

- Space Packet Protocol (reference [C1]);
- CCSDS File Delivery Protocol (CFDP) (reference [C2]);
- packet service as provided by the AOS Space Data Link Protocol (reference [C3]), TM Space Data Link Protocol (reference [C4]), and Unified Space Data Link Protocol (reference [C5]).

When transmission over a CCSDS space link occurs, application of one of the set of Channel Coding and Synchronization Recommended Standards will significantly reduce the loss of portions of transmitted data caused by data corruption.

Limits on the maximum size data unit that can be transmitted may be imposed by the protocol used or by other practical implementation considerations. The user is expected to take such limits into account when using this Recommended Standard.

3 INPUTS AND PARAMETERS

3.1 OVERVIEW

The compressor losslessly encodes a sequence of length-F binary input vectors $\boldsymbol{I}_t$, t = 0, 1,…. At each time index t, the compressor produces a variable-length binary output vector $\boldsymbol{o_t}$ that losslessly encodes $\boldsymbol{I}_t$. This encoding depends on several user-specified variables defined in 3.2:

– The *initial mask vector*, $\boldsymbol{M_0}$, is a length-F binary vector used to initialize the mask.

– The *minimum required effective robustness level*, R_t, controls the guaranteed number of consecutive output vectors that can be lost prior to the current output vector without affecting the ability to decompress it. The effective robustness level will be equal or greater than this value.

– The *new mask flag*, $\dot{p}_t$, when set to one, causes the current mask vector to be replaced with the mask being built. This enables bit positions to change classification from unpredictable to predictable if they have not changed state since the last time the new mask flag was set to one.

– The *uncompressed flag*, $\dot{r}_t$, causes the entire input vector to be included in the output vector when it is set to one; otherwise, only those bits classified as unpredictable are included.

– The *send mask flag*, $\dot{f}_t$, causes the entire mask vector to be encoded in the output vector when it is set to one.

All compressor parameters needed for decompression can be determined from the output binary vectors.

3.2 INPUT

At each time index t = 0, 1, …, the input to the compressor shall be a length-F binary input vector $\boldsymbol{I}_t$, where the input vector length F shall be a user-specified integer in the range $1 \leq F \leq 2^{16} - 1$.

3.3 PARAMETERS

3.3.1 The user-specified initial mask vector, $\boldsymbol{M_0}$, shall be a length-F binary vector.

NOTE – $\boldsymbol{M_0}$ could be derived from analysis of previous mask values or a stored value of the mask from the last time this was run. Setting $\boldsymbol{M_0} = \boldsymbol{0}$ (i.e., all positions predictable) is often a reasonable default.

3.3.2 In addition to the initial mask vector, compression parameters at each time index t consist of the following:

a) The user-specified minimum required effective robustness level, R_t, which shall be an integer between 0 and 7 at each time index *t*.

b) The user-specified new mask flag, $\dot{p}_t$, which shall be zero or one at each time index *t*.

c) The send mask flag, $\dot{f}_t$, which shall be $\dot{f}_t = 1$ for $t \leq R_t$; otherwise, $\dot{f}_t$ is user-specified and shall be zero or one.

d) The uncompressed flag, $\dot{r}_t$, which shall be $\dot{r}_t = 1$ if $t \leq R_t$; otherwise $\dot{r}_t$ is user-specified and shall be zero or one.

NOTE – The values of user-specified parameters $\boldsymbol{M_0}$, R_t, $\dot{f}_t$, $\dot{p}_t$, and $\dot{r}_t$ need not be known in advance, or communicated via means external to the compressor, for successful decompression.

4 MASK UPDATE

4.1 OVERVIEW

This section specifies the compressor's mask update stage. Two binary vectors, $\boldsymbol{M_t}$ (the mask) and $\boldsymbol{B_t}$ (the build), are updated in parallel based on the exclusive-or between the current input binary vector and the previous one (see 4.2.1 and 4.2.2).

Since no such difference is defined for the first input vector, build is initialized to a zero vector, that is, $\boldsymbol{B_0} = \boldsymbol{0}$. For subsequent input binary vectors, a bitwise XOR of sequential input binary vectors is used to update mask and build. The *change* vector, $\boldsymbol{D_t}$, is computed as the exclusive-or between the current mask and previous mask (see 4.2.3).

When the new mask flag, $\dot{p}_t$, is set to one, the mask is replaced by build, and build is reset to a zero vector; that is, all positions are classified as predictable (see 3.3.2). Hence, even though the update processes for mask and build depend on the same calculations, these vectors are not in general identical because they may be reset at different times during the encoding process.

4.2 MASK UPDATE

4.2.1 For $t \geq 0$, the build vector is a length-*F* binary vector defined as

$$\boldsymbol{B_t} = \begin{cases} (\boldsymbol{I_t} \text{ XOR } \boldsymbol{I_{t-1}}) \text{ OR } \boldsymbol{B_{t-1}}, & t > 0,\ \dot{p}_t = 0 \\ \boldsymbol{0}, & \text{otherwise} \end{cases}, \tag{6}$$

where $\dot{p}_t$ is defined in 3.3.2.

4.2.2 For $t > 0$, the mask vector is a length-*F* binary vector defined as

$$\boldsymbol{M_t} = \begin{cases} (\boldsymbol{I_t} \text{ XOR } \boldsymbol{I_{t-1}}) \text{ OR } \boldsymbol{M_{t-1}}, & \dot{p}_t = 0 \\ (\boldsymbol{I_t} \text{ XOR } \boldsymbol{I_{t-1}}) \text{ OR } \boldsymbol{B_{t-1}}, & \text{otherwise} \end{cases}, \tag{7}$$

where $\dot{p}_t$ is defined in 3.3.2.

NOTE – At $t = 0$, the initial mask vector $\boldsymbol{M_0}$ is user-specified (see 3.3.1).

4.2.3 For $t \geq 0$, the change vector $\boldsymbol{D_t}$ is a length-*F* binary vector defined as

$$\boldsymbol{D_t} = \begin{cases} \boldsymbol{M_t} \text{ XOR } \boldsymbol{M_{t-1}}, & t > 0 \\ \boldsymbol{0}, & \text{otherwise} \end{cases}. \tag{8}$$

5 ENCODER

5.1 OVERVIEW

This section specifies the encoding stage of the compressor. Subsection 5.2 defines functions used in the encoding process, and 5.3 specifies the calculation of the output vector from the input vector and compressor parameters.

As specified in 5.3, at any time, t, the encoder output, is a variable-length binary vector, $\boldsymbol{o_t}$, that is a concatenation of three variable-length binary vectors, $\boldsymbol{h_t}$, $\boldsymbol{q_t}$, and $\boldsymbol{u_t}$ described below.

The first binary vector, $\boldsymbol{h_t}$, encodes information about recent mask changes (see 5.3.3.1). The most recent change vector is ORed with previous change vectors (to provide robustness against packet loss), and the result is run-length encoded. Next, the effective robustness level, V_t, is encoded using 4 bits, followed by information on the mask values for each change. This is followed by the value of $\boldsymbol{c}_t$, which indicates if $\dot{p}_t$ was set to one more than once in the period covering this cycle and the previous V_t cycles. This information is used in the $\boldsymbol{u_t}$ encoding. The final bit indicates whether user-specified parameters $\dot{f}_t$ and $\dot{r}_t$ are both zero. If so, the values of those parameters are not encoded in $\boldsymbol{q_t}$ or $\boldsymbol{u_t}$.

The second binary vector, $\boldsymbol{q_t}$, encodes information about the entire mask (see 5.3.3.2). Although mask changes alone would be sufficient to reconstruct the mask, the option to send the entire mask can be requested by setting the send mask flag, $\dot{f}_t$, to one (see 3.3.2). In this case, $\boldsymbol{q_t}$ consists of a '1' concatenated with the mask that has been preprocessed and run-length encoded.

Finally, the third binary vector, $\boldsymbol{u_t}$, encodes different information depending on the values of $\dot{r}_t$ and $\boldsymbol{c}_t$. If $\dot{r}_t$ is one (see 3.3.2), then $\boldsymbol{u_t}$ always contains a bit string that encodes the value of the input block length F, followed by the input vector. Otherwise, if $\boldsymbol{c}_t$ is one, then $\boldsymbol{u_t}$ contains the unpredictable bit values and the values of any bits that changed from unpredictable to predicable in the present change vector or the previous V_t change vectors. If $\boldsymbol{c}_t$ is not one, then $\boldsymbol{u_t}$ contains only the unpredictable bit values. In all cases, the output is preceded by $\dot{r}_t$ if its value was not already specified by the last bit of $\boldsymbol{h_t}$.

5.2 BASIC ENCODING FUNCTIONS

5.2.1 OVERVIEW

This subsection defines the counter encoding, run-length encoding, and bit extraction functions used by the encoding procedure specified in 5.3.3.

5.2.2 COUNTER ENCODING FUNCTION

Given a positive integer $1 \leq A \leq 2^{16} - 1$, the counter encoding function, denoted COUNT(A), maps A onto a variable-length binary vector following table 5-1.

Table 5-1: Counter Encoding Table

Input Integer, A	*Output vector*
$A = 1$	'0'
$2 <= A <= 33$	'110' ‖ $\mathrm{BIT}_5(A-2)$
$A >= 34$	'111' ‖ $\mathrm{BIT}_E(A-2)$

E is calculated as

$$E = 2\lfloor \log_2(A-2) + 1 \rfloor - 6. \quad (9)$$

NOTE – The equation above calculates a bit string length consisting of a number of zeros concatenated with a minimum length bit string encoding of the integer $(\boldsymbol{A-2})$. As the relationship between the number of preceding zeros and the length of the bit string encoding $(\boldsymbol{A-2})$ is unique, it can be used by the decompressor to parse the output.

5.2.3 RUN-LENGTH ENCODING

Given a binary vector, $\boldsymbol{a}$, the run length encoding of $\boldsymbol{a}$, denoted RLE($\boldsymbol{a}$), is a variable-length binary vector defined as

$$\mathbf{RLE}(\boldsymbol{a}) = \mathrm{COUNT}(C_0) || \ldots || \mathrm{COUNT}\left(C_{\mathrm{H}(\boldsymbol{a})-1}\right) || \text{ '10' }, \quad (10)$$

where C_i is one more than the number of consecutive '0' bits preceding the ith '1' bit in $\boldsymbol{a}$, starting at the MSB and decreasing.

NOTES

1 If the vector $\boldsymbol{a}$ ends with one or more zeros, they are not explicitly encoded via the COUNT function, as they can be inferred from the length of the input vector and the number of '1' bits, H($\boldsymbol{a}$).

2 When vector $\boldsymbol{a}$ does not contain any '1' bits, the COUNT function will yield a '10' bit string.

3 Figure 5-1 provides an illustration of the run to count encoding.

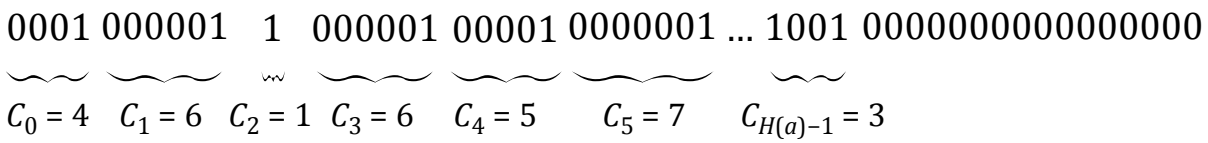

Figure 5-1: Example of the Run to Count Encoding, Converting a Binary Vector, $\boldsymbol{a}$, into a Sequence of Integers

5.2.4 BIT EXTRACTION FUNCTION

Given two binary vectors $\boldsymbol{a}$, $\boldsymbol{b}$, having the same length, the bit extraction of $\boldsymbol{a}$ relative to $\boldsymbol{b}$, denoted BE($\boldsymbol{a}$, $\boldsymbol{b}$), is the sequence of bits in $\boldsymbol{a}$ taken from the positions where $\boldsymbol{b}$ has a '1' bit. That is,

$$\mathrm{BE}(\boldsymbol{a},\boldsymbol{b}) = \dot{a}_{g_{\mathrm{H}(\boldsymbol{b})-1}} \| \cdots \| \dot{a}_{g_0}, \tag{11}$$

where g_i denotes the position of the ith '1' bit in $\boldsymbol{b}$, starting from the MSB.

5.3 ENCODING STEP

5.3.1 OUTPUT BINARY VECTOR STRUCTURE

The encoder output is comprised of the sequence of variable-length binary output vectors $\boldsymbol{o_0}$, $\boldsymbol{o_1}$, $\boldsymbol{o_2}$, $\boldsymbol{o_3}$, ...,. Each output binary vector $\boldsymbol{o_t}$ shall be defined as

$$\boldsymbol{o_t} = \boldsymbol{h_t} \| \boldsymbol{q_t} \| \boldsymbol{u_t}, \tag{12}$$

where variable-length binary vectors $\boldsymbol{h_t}$, $\boldsymbol{q_t}$, and $\boldsymbol{u_t}$ are specified in 5.3.3.1, 5.3.3.2, and 5.3.3.3, respectively. The calculation of these component vectors depends on intermediate calculations defined in 5.3.2.

5.3.2 INTERMEDIATE CALCULATIONS

5.3.2.1 For $t \geq 0$, $\dot{d}_t$ is defined as

$$\dot{d}_t = \begin{cases} 1, & \text{if } \dot{f}_t = 0 \text{ and } \dot{r}_t = 0 \\ 0, & \text{otherwise} \end{cases}, \tag{13}$$

where $\dot{f}_t$ and $\dot{r}_t$ are defined in 3.3.2.

5.3.2.2 For $t \geq 0$, V_t is defined as

$$V_t = \begin{cases} R_t, & \text{if } (t - R_t) \leq 0 \\ R_t + C_t, & \text{otherwise} \end{cases}, \tag{14}$$

where R_t is defined in 3.3.2, and C_t is the maximum integer where $C_t \leq min(t,\ 15) - R_t$, for which at given time instant t, all $\boldsymbol{D}_{t'} = \boldsymbol{0}$ (see 4.2.3) for $t' \in \{t - R_t - 1, t - R_t - 2, \ldots, t - R_t - C_t\}$.

NOTE – C_t counts the number of consecutive occurrences of no mask changes, starting from the first cycle not covered by the minimum required effective robustness level and working backwards in time.

5.3.3 OUTPUT VECTOR COMPONENTS

5.3.3.1 The binary vector $\boldsymbol{h_t}$ is defined as

$$\boldsymbol{h}_t = \ \mathrm{RLE}(\boldsymbol{X}_t) \parallel \mathrm{BIT}_4(V_t) \parallel \boldsymbol{e_t} \parallel \boldsymbol{k}_t \parallel \boldsymbol{c}_t \parallel \dot{d}_t, \tag{15}$$

where V_t is defined in 5.3.2.2, $\dot{d}_t$ is defined in 5.3.2.1, and $\boldsymbol{X}_t$, $\boldsymbol{e}_t$, $\boldsymbol{c}_t$, and $\boldsymbol{k}_t$ are defined as

$$\boldsymbol{X}_t = \begin{cases} < \boldsymbol{D}_t >, & \text{if } R_{t_t} = 0 \\ < (\boldsymbol{D}_1 \text{ OR } \boldsymbol{D}_2 \text{ OR} \ldots \text{OR } \boldsymbol{D}_t) >, & \text{if } (t - R_t) \leq 0 \\ < (\boldsymbol{D}_{t-R_t} \text{ OR } \boldsymbol{D}_{t-R_t+1} \text{ OR} \ldots \text{OR } \boldsymbol{D}_t) >, & \text{otherwise} \end{cases} \tag{16}$$

where $\boldsymbol{D_t}$ is the change binary vector defined in 4.2.3, and R_t is defined in 3.3.2,

$$\boldsymbol{y}_t = \mathrm{BE}(< \sim \boldsymbol{M}_t >, \boldsymbol{X}_t), \tag{17}$$

$$\boldsymbol{e}_t = \begin{cases} \emptyset, & \text{if } V_t = 0 \text{ or } \boldsymbol{X}_t = \boldsymbol{0} \\ \text{'0'}, & \text{if } \boldsymbol{y}_t = \boldsymbol{0} \text{ and } V_t > 0 \text{ and } \boldsymbol{X}_t \neq \boldsymbol{0} \\ \text{'1'}, & \text{otherwise} \end{cases}, \tag{18}$$

$$\boldsymbol{k}_t = \begin{cases} \emptyset, & \text{if } V_t = 0 \text{ or } \boldsymbol{X}_t = \boldsymbol{0} \text{ or } \boldsymbol{y}_t = \boldsymbol{0} \\ \boldsymbol{y}_t, & \text{otherwise} \end{cases}, \tag{19}$$

$$\boldsymbol{c}_t = \begin{cases} \emptyset, & \text{if } \boldsymbol{k}_t = \emptyset \\ \text{'0'}, & \text{if } \boldsymbol{k}_t \neq \emptyset \text{ and } \dot{p}_i \text{ is not set to one more than once, for} \\ & i \text{ in } \{\max(0, t - V_t), \max(0, t - V_t) + 1, \ldots, t - 1, t\}, \\ \text{'1'}, & \text{otherwise} \end{cases}. \tag{20}$$

5.3.3.2 The binary vector $\boldsymbol{q_t}$ is defined as

$$\boldsymbol{q_t} = \begin{cases} \emptyset, & \text{if } \dot{d}_t = 1 \\ \text{'1'} \parallel \mathrm{RLE}\big(< (\boldsymbol{M_t} \text{ XOR } (\boldsymbol{M_t}^{\ll})) >\big), & \text{if } \dot{f}_t = 1 \\ \text{'0'}, & \text{otherwise} \end{cases}, \tag{21}$$

where $\dot{f}_t$ is defined in 3.3.2, and $\dot{d}_t$ is defined in 5.3.2.1.

5.3.3.3 The binary vector $\boldsymbol{u_t}$, is defined as

$$\boldsymbol{u_t} = \begin{cases} \mathrm{BE}(\boldsymbol{I_t}, (< \boldsymbol{X}_t > \ OR\ \boldsymbol{M_t})), & \text{if } \dot{d}_t = 1 \text{ and } \boldsymbol{c}_t = 1 \\ \mathrm{BE}(\boldsymbol{I_t}, \boldsymbol{M_t}), & \text{if } \dot{d}_t = 1 \text{ and } \boldsymbol{c}_t \neq 1 \\ '1' \parallel \mathrm{COUNT}(F) \parallel \boldsymbol{I_t}, & \text{if } \dot{r}_t = 1 \\ '0' \parallel \mathrm{BE}\big(\boldsymbol{I_t}, (< \boldsymbol{X}_t > \ OR\ \boldsymbol{M_t})\big), & \text{if } \dot{r}_t = 0 \text{ and } \dot{f}_t = 1 \text{ and } \boldsymbol{c}_t = 1 \\ '0' \parallel \mathrm{BE}(\boldsymbol{I_t}, \boldsymbol{M_t}), & \text{otherwise,} \end{cases} \tag{22}$$

where $\dot{r}_t$ and $\dot{f}_t$ are defined in 3.3.2, $\boldsymbol{c}_t$ is defined in 5.3.3.1, and $\dot{d}_t$ is defined in 5.3.2.1.

ANNEX A

IMPLEMENTATION CONFORMANCE STATEMENT (ICS) PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Implementation Conformance Statement (ICS) Requirements List (RL) for an implementation of *Robust Compression of Fixed-Length Housekeeping Data*, CCSDS 124.0-R-1, June 2022. The ICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation claiming conformance must satisfy the mandatory requirements referenced in the RL.

A1.2 ABBREVIATIONS AND CONVENTIONS

The RL consists of information in tabular form. The status of features is indicated using the abbreviations and conventions described below.

Item Column

The label in the item column identifies the item in the table.

The use of nested item labels indicates subordination of conditional items. For example, an item with label L*i*.*j* is not applicable unless the parent item L*i* is supported.

Description Column

The description column contains a brief description of the item. It implicitly means 'Is this item supported by the implementation?'

Reference Column

The reference column indicates the relevant subsection of *Robust Compression of Fixed-Length Housekeeping Data*, CCSDS 124.0-R-1 (this document).

Status Column

The status column uses the following notations:

M mandatory.

O optional.

N/A not applicable.

O.*i* qualified optional—for a group of related optional items labeled by the same numeral *i*, it is mandatory to support at least one of the items.

C.*j* conditional—the requirement on the capability ('M', 'O', or 'N/A') depends on the support of another optional item. The numeral *j* identifies a unique conditional status expression defined immediately following the table.

C:<status> indicates that the status applies for the given subordinate item when the parent item is supported and is not applicable otherwise.

<condition>:<status> indicates that the status applies only when the given condition is met, and is not applicable otherwise. For example, '(Q2 or Q3):M' indicates that support for the item is mandatory if item Q2 or item Q3 are supported and is not applicable otherwise.

Values Allowed Column

The values allowed column contains the list or range of values allowed. The following notations are used:

range of values:	<min value> .. <max value>
example:	2 .. 16
list of values:	<value1>, <value2>, ..., <valueN>
example:	3, 6, 9, …, 21
N/A	not applicable

Item Support or Values Supported Column

In the item support or values supported column, the support of every item as claimed by the implementer shall be stated by entering the appropriate answer ('Y', 'N', or 'N/A') or the values supported:

Y yes, item supported by the implementation.

N no, item not supported by the implementation.

N/A not applicable.

Prerequisite Line

A prerequisite line takes the form: Prerequisite: <predicate>. A prerequisite line at the top of a table indicates that the table need not be completed if the predicate is FALSE.

A1.3 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the Recommended Standard by completing the RL; that is, the state of compliance with all mandatory requirements and the options supported are shown. The resulting completed RL is called an ICS. The implementer shall complete the RL by entering appropriate responses in the support or values supported column, using the notation described in A1.2. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference X*i*, where *i* is a unique identifier, to an accompanying rationale for the noncompliance.

A2 ICS PROFORMA FOR ROBUST COMPRESSION OF FIXED LENGTH HOUSEKEEPING DATA

A2.1 GENERAL INFORMATION

A2.1.1 Identification of ICS

Date of Statement (DD/MM/YYYY)	
ICS serial number	
System Conformance statement cross-reference	

A2.1.2 Identification of Implementation Under Test

Implementation Name	
Implementation Version	
Function Implemented	Compression_____ Decompression_____
Special Configuration	
Other Information	

A2.1.3 Identification of Supplier

Supplier	
Contact Point for Queries	
Implementation Name(s) and Versions	
Other information necessary for full identification, for example, name(s) and version(s) for machines and/or operating systems; System Name(s)	

A2.1.4 Identification of Specification

CCSDS 124.0-R-1 Issue 1	
Have any exceptions been required? NOTE – A YES answer means that the implementation does not conform to the Recommended Standard. Non-supported mandatory capabilities are to be identified in the ICS, with an explanation of why the implementation is non-conforming.	Yes [ ] No [ ]

A2.2 REQUIREMENTS LIST

A2.2.1 Requirements

Table A-1: Requirement List

Item	Description	Reference	Status	Support
R1	Input	3.2	M	
R2	Parameters	3.3	M	
R3	Mask Update	4.2	M	
R4	Encoding Functions	5.2	M	
R5	Encoding Step	5.3	M	

A2.2.2 Parameters

Table A-2: Parameters

Item	Description	Reference	Status	Values Allowed	Item Support or Values Supported
P1	Input Vector Length, F supported value.	3.2	M	$1 .. 2^{16} - 1$.	
P2	Configuration of initial mask vector, $\boldsymbol{M_0}$	3.3.1	M	$0 .. 2^{F}-1$	

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

B1.1 INTRODUCTION

This Recommended Standard assumes that any required security is provided by encryption, authentication methods, and access control that will be performed in other layers, for example, Application, Transport, Data Link. Mission and service providers are expected to select from recommended security methods suitable to the specific application profile. Specification of these security methods and other security provisions is outside the scope of this Recommended Standard.

B1.2 SECURITY CONCERNS WITH RESPECT TO THE CCSDS DOCUMENT

Security concerns in the areas of data privacy, integrity, authentication, access control, availability of resources, and auditing are to be addressed in the appropriate layers and are not specified in this Recommended Standard. The use of lossless data compression does not affect the proper functioning of methods used to achieve such protection. The use of lossless data compression slightly improves data integrity because the alteration of even a single bit of compressed data is likely to cause conspicuous and easily detectible corruption of the reconstructed data, thus making it more likely that malicious data alteration will be detected. Also, the use of compression improves randomness/entropy of plaintext data by removing data redundancy. Hence, it is recommended to apply security after compression and not before.

For this reason, data compression algorithms are a well-known option for whitening the output of an imperfect random bit generation function.

B1.3 POTENTIAL THREATS AND ATTACK SCENARIOS

An eavesdropper will not be able to decompress compressed data if proper encryption is performed at a lower layer.

B1.4 CONSEQUENCES OF NOT APPLYING SECURITY TO THE TECHNOLOGY

There are no specific security measures prescribed for compressed data. Therefore, consequences of not applying security are only imputable to the lack of proper security measures in other layers. A distinction can be made between encryption and authenticated encryption. Decompression will fail if there are undetected errors after decryption. The cause of undetected errors could be random noise combined with undetected error performance of synchronization and channel coding functions and/or malicious data manipulation. The highest integrity protection can only be achieved with the use of authenticated encryption on top of communication protocol integrity mechanisms (e.g., channel coding, frame integrity).

B2 SANA CONSIDERATIONS

The recommendations of this document do not require any action from SANA. All configuration parameter values needed for decompression can be read from the compressed bitstream.

B3 PATENT CONSIDERATIONS

Implementers should be aware that the algorithm and compressed data format defined herein are the subject of a patent application with no. EP3580902B1 protected by the European Space Agency. The European Space Agency has provided a signed ISO Patent Statement and Licensing Declaration Form with a declaration that it is prepared to grant a free-of-charge license to an unrestricted number of applicants on a worldwide, non-discriminatory basis and under other reasonable terms and conditions to make, use, and sell implementations of this patent.

Potential user agencies should direct their requests for licenses to:
Aude de Clercq
Technology Transfer and Patent Office
European Space Agency
Keplerlaan 1
2201 AZ Noordwijk
The Netherlands
Tel: +31 71 565 8662
E-mail: Patent@esa.int

ANNEX C

INFORMATIVE REFERENCES

(INFORMATIVE)

[C1] *Space Packet Protocol*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 133.0-B-2. Washington, D.C.: CCSDS, June 2020.

[C2] *CCSDS File Delivery Protocol (CFDP)*. Issue 5. Recommendation for Space Data System Standards (Blue Book), CCSDS 727.0-B-5. Washington, D.C.: CCSDS, July 2020.

[C3] *AOS Space Data Link Protocol*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.0-B-4. Washington, D.C.: CCSDS, October 2021.

[C4] *TM Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 132.0-B-3. Washington, D.C.: CCSDS, October 2021.

[C5] *Unified Space Data Link Protocol*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.1-B-2. Washington, D.C.: CCSDS, October 2021.

[C6] David Evans, et al. "Implementing the New CCSDS Compression Standard (POCKET+)—The Best of Two Worlds." In *Proceedings of the 16th International Conference on Space Operations, SpaceOps-2021 (3–5 May 2021, Cape Town, South Africa)*. Paris: International Astronautical Federation, 2021.

ANNEX D

ABBREVIATIONS AND ACRONYMS

(INFORMATIVE)

Term	Meaning
CCSDS	Consultative Committee on Space Data Systems
CFDP	CCSDS File Delivery Protocol
ICS	Implementation Conformance Statement
LSB	least significant bit
MSB	most significant bit
RL	requirements list