



Interface Control Document (ICD)

I. Scope

SEDAP-Express is an exceptionally fast path to integrate new applications, sensors, effectors or other similar things into the ecosystem of MESE. That's why it is intentionally kept simple and offers several technical ways of communication. Of course, this results in limitations, but in most cases where quick and easy integration is required, these are negligible. If increased demands arise later on, the "bigger" SEDAP API respective MESE interface can be used if necessary. SEDAP-Express is licensed under the "Simplified BSD License" (BSD-2-Clause). Therefore, there should be no problems using SEDAP-Express in commercial or non-commercial projects or integrating parts of the SEDAP-Express framework. Everything you need for development and testing can be found on the Internet at <https://SEDAP.Express>.

II. Glossary

| | | |
|-------------|---|--|
| MESE | = | Military Expandable Software Environment |
| SEDAP | = | Safety critical Environment for Data exchange And Process scheduling |
| CSV | = | Comma-Separated-Values |
| SEC | = | SEDAP-Express-Connector |
| SECMockUp | = | Simulation of the real SEC and a C2-like system with a simple map |
| MessageTool | = | Tool for manual generation of messages |
| SIDC | = | Symbol identification code (APP-6A/B/MIL-STD-2525B/C/STANAG 2019) |
| ASCII | = | American Standard Code for Information Interchange – in this context the ISO-8859-1 table is meant |
| Base64 | = | Binary-to-text encoding scheme, which is using an alphabet of 64 characters |

III. General connection attributes

1. Common conventions

- Basic format is CSV using ';' (0x3B) as separation character between two data fields
- Message ends with a '\n' (0x0A) as termination
- Elements of list values are separated by '#' (0x23), List values are marked with a '*' (
- Mandatory fields/elements except the name are marked with (M)
- The messages are human-readable and using the ASCII-table
- Fields with (Binary)Data which possibly contains a special character (e.g. 0x0A, 0x23, 0x3B) has to be encoded with Base64
- Unknown/Invalid values must not be transmitted, the respective field will be left empty
- If there are only ';' (0x3B) characters left in the message, these can be cut off
- Support for IPv4 or IPv6 (except for serial connection)
- SEC/SECMockUp/Applications can send and receive at any time
- Application shall send heartbeat message not more often than with 1Hz (+-100ms), but can vary if it is required
- SEC/SECMockUp answers heartbeat also with a heartbeat message (see chapter IV.2.12)

2. Authentication

- If authentication/encryption is required, it's always preferred to use VPN connections instead
- Messages could be authenticated by fulfilling the MAC field using a shared password (see chapter IV.1.1)
- The password can either be defined in advance or exchanged using the Diffie-Hellman process and the KEYEXCHANGE message (see chapter IV.2.13). It's recommended to authenticate even this message with a pre-shared password.
- For calculating the MAC you have to use the complete message incl. header and setting the MAC temporary to "0000"
- To save data, it is possible to limit the MAC to the first 4 bytes/32 bits while reducing security
- At minimum standards defined by FIPS 140-2 (Federal Information Processing Standard) have to be used
That means it is preferred to use:
 - o 32Bit/256Bit HMAC in combination SHA256 (FIPS 198-1 / RFC 2104)
 - o 32Bit/128Bit CMAC in combination AES128 (NIST SP 800-38B)
 - o 32Bit/128Bit GMAC in combination AES128 (NIST SP 800-38D)
- It's recommended to use MAC DBRG (Deterministic Random Bit Generator, NIST SP 800-90A/B)

Sample (32Bit CMAC, Password:expressexpressex): OWNUNIT;5E;661D4410;66A3;R;;;53.32;8.11;0;5.5;21;22;;;FGS Bayern;sfsfpclff-----

3. Encryption

- Encryption is optional and as already wrote, if authentication/encryption is required, it's preferred to use VPN instead
- At minimum standards defined by FIPS 140-2 (Federal Information Processing Standard) have to be used
That means it is preferred to use:
 - o AES128/256 CFB/NoPadding (NIST SP 800-38A)
 - o AES128/256 CTR/NoPadding (NIST SP 800-38A)
 - o XOR (Pseudo-encryption ONLY for testing/debugging purposes or for very light obfuscation)
- It's recommended to use MAC DBRG (Deterministic Random Bit Generator, NIST SP 800-90A/B)
- Encrypted data must be Base64 encoded - all incl. header (see chapter IV.1.1.1) have to be encoded
- If there is password given every message have to be encrypted – mixture of encrypted and plain message is not allowed

Sample reference message:

OWNUNIT;5E;661D4410;66A3;R;;;53.32;8.11;0;5.5;21;22;;;FGS Bayern;sfsplclff——

Sample encrypted message (AES128, ECB, Password:expressexpressex):

SpkxMb4T08Py6MDfwRJUyIJLE45edpyrZ3pFSw0vWdbk/Ry1RKeSx1gFCpzGhVLsfx0iNQ6fuUwtG9UfweXRSvN5Lk0XMN6TYAc4TOHosOI=

4. Compression

- Compression is optional (has to be used BEFORE an optional encryption because of the high entropy)
- Use the widely spreaded “deflate” for compression (only effective if the message size exceeds 140 characters)
- Compressed data must be Base64 encoded – all incl. header (see chapter IV.1.1.1) have to be compressed
- If the first bytes of a received message doesn't match a message name, prove for compression

Sample reference message:

TEXT;czG1NjMzdDEzNT01NjYycbU0trY2slYKycgsVgCiRIW8/JLM5FRFJQA=

Sample compressed message:

eJwLcY0IsXYxsZyZM3QxNzA2sjY2MnG1Dra2NrZWCsnILFYAokSF5IzEEoXc1OLixPRURSVrVxM3QwC2Vg/Y

5. TCP Connection

- Standard port 50000, but customizable
- SEC/SECMockUp can play server or client role
- One should use only one TCP connection for any kind of message and keep it open

6. UDP-Connection

- Standard port 50000, but customizable
- Support for Uni-, Broad- or Multicast mode
- Standard Multicast-Address is 228.2.19.80 resp. ff02:8:2:19:80::1
- Multiple messages per UDP-packet possible
- SEC/SECMockUp answers heartbeat message with UDP-Unicast (see chapter IV.2.12)

7. Serial connection

- Standard 115200-8-N-1, Full-Duplex is preferred, Half-Duplex and Simplex without acknowledge requests
- There are three modes available, depending on the use case:
 - o Message never contains a \n (0x0A), therefore the message could be sent with an additional appended \n\n\n\n
 - o Message contains one or more \n (0x0A), have to be Base64-encoded and sent with an additional appended \n\n\n\n

8. REST-API

- Standard ports are HTTP 80 or HTTPS 443, but customizable
- Deflate or gzip compression should be supported

9. Protocol Buffers

- Ports and other TCP or UDP parameters are the same as described in chapter III.5 and III.6.

IV. Data Exchange between SEC/SECMockUp and client applications

1. General

1.1. TCP-/UDP-/Serial connections

On principle, messages have a CSV structure with a common header:

<Name>(M);<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;<Content>

Everything except the name is basically optional. Certain messages require specific header elements, such as time or the sender. These exceptions are described in the chapter of the respective message. Hexadecimal numbers have no "0x"-prefix.

| | |
|-------------------|--|
| <Name> | Defines the purpose of a message. Sometimes it's so-called topic. |
| <Number> | This is a hexadecimal string representation of an 8-bit sequential number. Each type of message has its own counter that starts again with zero after reaching 255/FF. A reconnect resets the counters! |
| <Time> | A hexadecimal string representation of a 64-bit Unix time stamp with milliseconds. |
| <Sender> | In most cases, you should use a hexadecimal string representation of a 16-bit unsigned integer, but you can also use freely selected textual identifiers. This field won't be changed, even if a message has been forwarded or relayed. This sender identification can be chosen randomly by the participants themselves or permanently assigned by a responsible institution when preparing a specific use/network. If information of a sub-system has to be forwarded the sender identification should be the source of the original information, in that case of the sub-system. |
| <Classification> | Describes the classification or security level of the content. Possible values are P=public, U=unclassified, R=restricted, C=confidential, S=secret, T=top secret |
| <Acknowledgement> | TRUE=request an acknowledgement, FALSE/Nothing=No acknowledgement |
| <MAC> | Hash-based message authentication code for verification |
| <Content> | Content of the message, depending on the message purpose. |

1.2. REST-API connection

If the REST-API shall be used, it's preferred to also use the provided JSON schema file and the generated code which either comes with the SEDAP-Express SDK or has been generated by yourself. You can find the schema in chapter IV.3 or on <http://sepap.express>.

1.3. Protobuf connection

At least it's also possible to use Google™ protocol buffers to exchange the SEDAP-Express messages. You can find the schema in chapter IV.4 or on <http://sepap.express>. This allows you to generate your own code or use the existing code from the framework or the sample client.

2. Messages

This is the list of all so-far available messages and their structures and content including some samples. All units of measurement are generally given in SI-units, but there are deviations where this makes sense due to the usual range of values. In the following the used units will be given within square brackets for all message-descriptions. The altitude is the altitude above sea-level. A value of zero means exactly on ground, if the position is on land. Latitude and longitude are in decimal degrees, while positive values means north and east respective negative values south and west. Relative position values are defined this way, that the x-axis points to the east direction, y-axis points to the north and the z-axis is equal to the height above the unit. Speed and course are meant to be relative to ground. Course and heading have a range from zero to 359.9999, are relative to geographic north (zero degree) and rotate clockwise. In general, all values are optional. mandatory parameters are marked with (M). In general, numerical values are rational numbers (floating point), unless otherwise defined. In the examples, fields are sometimes intentionally shortened because they would otherwise be too large. This is the case, for example, with the KEYCHANGE message.

2.1. OWNUNIT

Description: Positional, kinematic and identification data of the own (sent by the client) or host (sent by the SEC) unit/platform. If a client is sending this message, it will be converted to a contact and sent into the MESE network.

Structure:

OWNUNIT;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<Latitude>[°](M);<Longitude>[°](M);<Altitude>[m];
<Speed over ground>[m/s];<Course over ground>[°];
<Heading>[°];<Roll>[°];<Pitch>[°];
<Name>;<SIDC>

Sample 1: OWNUNIT;5E;661D4410;66A3;R;;;53.32;8.11;0;5.5;21;22;;;FGS Bayern;sfspfclff-----

Sample 2: OWNUNIT;5E;661D4410;66A3;R;TRUE;;42.32;-123.11;10000;50.23;297;;;33.3;-0.15;sfapmf-----

2.2. CONTACT

Description: Positional, kinematic and identification data of a contact. For example, this message would be used by a sensor to report a contact it recognized. In return this message would be used to receive the tactical picture from the MESE network.

Structure:

CONTACT;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<ContactID>(M);<DeleteFlag>;<Latitude>[°](M);<Longitude>[°](M);<Altitude>[m];
<relative X-Distance>[m];<rel Y-Distance>[m];<rel Z-Distance>[m];
<Speed over ground>[m/s];<Course over ground>[°];
<Heading>[°];<Roll>[°];<Pitch>[°];
<width>[m];<length>[m];<height>[m];
<Name>;<Source>;<SIDC>;<MMSI>;<ICAO>;<Image>;<Comment>

| | | |
|------------|--------|--|
| ContactID | ASCII | A positive identification unique number or free text of the contact chosen by the sender of this message |
| DeleteFlag | TRUE | Contact has to be removed |
| | FALSE | Contact is current |
| Source | ASCII | Available types (more than one available): R=Radar, A=AIS, I=IFF/ADS-B, S=Sonar, E=EW, O=Optical, Y=Synthetic, M=Manual |
| SIDC | SIDC | Identification code |
| MMSI | MMSI | Maritime Mobile Service Identity |
| ICAO | ICAO | International Civil Aviation Organization |
| Image | Base64 | Imagedata (JPG, PNG, TIF) encoded in Base64 |
| Comment | ASCII | Free text to the contact |

Sample 1: CONTACT;5E;661D4410;66A3;R;;;100;FALSE;53.32;8.11;0;5.5;21;22;;;;;FGS Bayern;AR;sfspfc1ff-----;;Ch22

Sample 2: CONTACT;5F;661D5420;66A3;U;;;101;FALSE;36.32;12.11;2000;44;331;;11;65;10;Unknown;O;;22113321;;NL

Sample 3: CONTACT;60;661B7410;66A3;S;TRUE;;102;TRUE;53.32;8.11

2.3. EMISSION

Description: Positional, attributes and identification data of an electro-magnetic, optical or acoustic emission.

Structure:

EMISSION;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<EmissionID>(M);<DeleteFlag>;<SensorLatitude>[°](M);<SensorLongitude>[°](M);<SensorAltitude>[m];
<EmitterLatitude>[°];<EmitterLongitude>[°];<EmitterAltitude>[m];
<Bearing>[°](M);<Frequencies[Hz]*>;<Bandwidth[Hz]>;<Power[db(A)]>;<FreqAgility>;<PRFAgility>;
<Function>;<SpotNumber>;<SIDC>;<Comment>

| | | |
|-------------|----------|--|
| EmissionID | Number>0 | A positive identification unique number of the emission chosen by the sender of this message. This number should also be unique in terms of contact numbers. |
| DeleteFlag | TRUE | Emission has to be removed |
| | FALSE | Emission is current |
| FreqAgility | 00 | Stable Fixed |
| | 01 | Agile |
| | 02 | Periodic |
| | 03 | Hopper |
| | 04 | Batch Hopper |
| | 05 | Unknown |
| PRFAgility | 00 | Fixed periodic |
| | 01 | Staggered |
| | 02 | Jittered |
| | 03 | Wobulated |
| | 04 | Sliding |
| | 05 | Dwell switch |
| | 06 | Unknown PRF |
| | 07 | CW |
| Function | 00 | Unknown |
| | 01 | ESM Beacon/Transponder |
| | 02 | ESM Navigation |
| | 03 | ESM Voice Communication |
| | 04 | ESM Data Communication |

| | | |
|---------|-------|-------------------------------|
| | 05 | ESM Radar |
| | 06 | ESM IFF/ADS-B |
| | 07 | ESM Guidance |
| | 08 | ESM Weapon |
| | 09 | ESM Jammer |
| | 0A | ESM Natural |
| | 0B | ACOUSTIC Object |
| | 0C | ACOUSTIC Submarine |
| | 0D | ACOUSTIC Variable Depth Sonar |
| | 0E | ACOUSTIC Array Sonar |
| | 0F | ACOUSTIC Active Sonar |
| | 10 | ACOUSTIC Torpedo Sonar |
| | 11 | ACOUSTIC Sono Buoy |
| | 12 | ACOUSTIC Decoy Signal |
| | 13 | ACOUSTIC Hit Noise |
| | 14 | ACOUSTIC Propeller Noise |
| | 15 | ACOUSTIC Underwater Telephone |
| | 16 | ACOUSTIC Communication |
| | 17 | ACOUSTIC Noise |
| | 18 | LASER Range Finder |
| | 19 | LASER Designator |
| | 1A | LASER Beam Rider |
| | 1B | LASER Dazzler |
| | 1C | LASER Lidar |
| SIDC | SIDC | Identification code |
| Comment | ASCII | Free text to the emission |

Sample 1: EMISSION;5E;661D4410;66A3;R;;;100;;53.32;8.11;0;;;20;8725000#8735000;20000;3;0;2;6;5;10233;SA-8

Sample 2: EMISSION;5F;661D5410;66A3;R;;;101;;54.86;9.32;0;52.12;9.80;50;233;25725;4000;1,5;2;0;6;;sngpesr-----

2.4. METEO

Description: Metrological data of the environment.

Structure:

METEO;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>; <MAC>;
<SpeedThroughWater>[m/s];<WaterSpeed>[m/s];<WaterDirection>[°];<WaterTemperature>[°C];<WaterDepth>[m];
<AirTemperature>[°C];<DewPoint>[°C];<HumidityRel>[%];<Pressure>[hPa];<WindSpeed>[m/s];<WindDirection>[°];
<Visibility>[km];<CloudHeight>[m];<CloudCover>[%]

Sample: METEO;AC;661D44C0;74BE;U;;;15.4;15.5;;;10.2;72;20.3;;55;1005;25;;;2500;33

2.5. TEXT

Description: Human readable textual data. This could be an alert message, but also a simple text message for chatting. If the text possibly contains special characters (e.g. UTF-x, 0x0A, 0x23, ... see chapter III.1), it has to be Base64-encoded and the encoding indicator has to be set. If the coding indicator is not set, no coding is assumed.

Structure:
TEXT;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<Type>;<Encoding>;<Text>(M);<Recipient>

| | | |
|-----------|-----------|---|
| Type | 00 | Undefined |
| | 01 | Alert |
| | 02 | Warning |
| | 03 | Notice |
| | 04 | Chat |
| Encoding | BASE64 | Text is Base64 encoded |
| | NONE | Text is not encoded |
| Text | ASCII | Free text of the message |
| Recipient | HexString | In most cases, you should use a hexadecimal string representation of a 16-bit unsigned integer, but you can also use freely selected textual identifiers, as explained in the table from chapter IV.1.1 |

Sample 1: TEXT;D3;661D44D2;324E;S;TRUE;;1;NONE;"This is an alert!"
Sample 2: TEXT;D4;661D458E;324E;S;TRUE;;2;NONE;"This is a warning!"
Sample 3: TEXT;D5;661D6565;324E;S;;;3;;"This is a notice!"
Sample 4: TEXT;D6;661D7032;324E;S;;;4;BASE64;IIRoaxMgaXMgYsBjaGF0IG1lc3NhZ2Uhlg==;E4F1

2.6. GRAPHIC

Description: Define graphical plans likes polygons, squares or routes

Structure:

GRAPHIC;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<GraphicType>(M);<LineWidth>;<LineColor>;<Annotation>;<additional GraphicType-dependent parameters>*

| | | |
|-------------|--------|---|
| GraphicType | 00 | Point: <Latitude>[°];<Longitude>[°];<Altitude>[m] |
| | 01 | Path: <Latitude>[°];<Longitude>[°];<Altitude>[m] # ... |
| | 03 | Polygon: <Latitude>[°];<Longitude>[°];<Altitude>[m] # ... |
| | 04 | Rectangle: <RotationAngle>[°];<Latitude1>[°];<Longitude1>[°];<Altitude1>[m]#<Latitude2>[°],<Longitude2>[°];<Altitude2>[m] |
| | 05 | Square: <Latitude>[°];<Longitude>[°];<Altitude>[m];Radius-X[m];Radius-Y[m] |
| | 06 | Circle: <Radius>[m];<Latitude>[°];<Longitude>[°];<Altitude>[m] |
| | 07 | Ellipse: <Radius-X>[m];<Radius-Y>[m];<CenterLatitude>[°];<CenterLongitude>[°];<CenterAltitude>[m] |
| | 08 | Block: <Latitude>[°];<Longitude>[°];<Altitude>[m];X-Radius [m];Y-Radius [m];Z-Radius [m] |
| | 09 | Sphere: <Latitude>[°];<Longitude>[°];<Altitude>[m];Radius[m] |
| | 0A | Ellipsoid: <Center_Latitude>[°];<Center_Longitude>[°];<Center_Altitude>[m];<Radius-X>[m];<Radius-Y>[m];<Radius-Z>[m] |
| LineWidth | => 1 | Width of the line or the point |
| LineColor | RGBA | Color of the line or the point in Web notation 800000FF for a darker red |
| FillColor | RGBA | Color of the line or the point in Web notation 00FF0080 for translucent green |
| Encoding | BASE64 | Text is Base64 encoded |
| | NONE | Text is not encoded |
| Annotation | ASCII | Text for an annotation to this graphic |

Sample 1: GRAPHIC;79;661D62C0;910E;U;;;8;1;FF8000;BASE64;QXJIYSBBbHBoYQ==;10000;53.43;9.45

Sample 2: GRAPHIC;78;661D64C0;910E;U;;;1;1;808080;;Transit;54.23,12.86#54.30,12.9#54.55,13.3

2.7. COMMAND

Description: Command for one specific or all possible recipients. Which camera is assigned to which number and which camera modes are available must be defined specifically for each application and depending on the sensor platform. The same also applies in particular for the generic action and any additional parameters for it. Time stamps are optional and in milliseconds. If no time stamp is given, it's interpreted as instantly. Of course, you cannot use the generic SEDAP Express connector in these cases, but must implement a specific and user-defined connector.

Structure:

COMMAND;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<Recipient>(M);<CmdID>;<CmdType>(M);<additional cmdType-dependent parameters>*

| | | |
|-----------|-----------|---|
| Recipient | HexString | In most cases, you should use a hexadecimal string representation of a 16-bit unsigned integer, but you can also use freely selected textual identifiers. |
| CmdID | HexString | Hexadecimal identifier of the command |
| CmdType | 00 | Power off: <Unix time stamp> |
| | 01 | Restart: <Unix time stamp> |
| | 02 | Standby: <Unix time stamp> |
| | 03 | Wake up: <Unix time stamp> |
| | 04 | Sync time: <IP/Hostname of a NTP server> |
| | 05 | Send status |
| | 06 | Move: <Latitude>[°];<Longitude>[°];<Altitude>[m] |
| | 07 | Rotate: <RotationAngle>[°] |
| | 08 | Scan Area: <Latitude1>[°];<Longitude1>[°];<Latitude2>[°];<Longitude2>[°];<RotationAngle>[°] |
| | 09 | Take photo: <Number of camera>;<Camera mode> |
| | 0A | Make video: <Number of camera>;<Camera mode>;<Duration> |
| | 0B | Switch on live video stream: <Number of camera>;<Camera mode> |
| | 0C | Switch off live video stream: <Number of camera> |
| | 0D | Engagement: start-engagement hold-engagement stop-engagement;<contactID> |
| | FF | Generic Action: <Kind of action>(has to be defined individually, inclusive implementing an custom connector software) |

Sample 1: COMMAND;27;661D44C0;E4B3;C;TRUE;;AB49;13;hold-engagement;1000

Sample 2: COMMAND;29;661D44C0;E4B3;C;TRUE;;Drone1;FF;OPEN_BAY

2.8. STATUS

Description: This message transports some kinds of status variables like the remaining batterie capacity and optionally the execution status of the last or a specific command (see chapter IV.2.7). If the status information of a sub-system has to be forwarded, the sender identification should be the source of the original information, meaning for swarm of drones, the concrete, individual drone identification.

Structure:

STATUS;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<TecStatus>;<OpsStatus>;<AmmunitionLevel>;<FuelLevel>;<BatterieLevel>;<CmdState>;<CmdID>;<IP/Host>;<Media>;<Encoding>;<Text>

| | | |
|-----------------|-----------|--|
| TecStatus | 0 | Off/Absent |
| | 1 | Initializing |
| | 2 | Degraded |
| | 3 | Operational |
| | 4 | Fault |
| OpsStatus | 0 | Not operational |
| | 1 | Degraded |
| | 2 | Operational |
| AmmunitionLevel | % | Relative remaining ammunition |
| FuelLevel | % | Relative remaining fuel capacity |
| BatterieLevel | % | Relative remaining batterie capacity |
| CmdState | 01 | Executed successfully |
| | 02 | Partially executed successfully |
| | 03 | Executed not successfully |
| | 03 | Execution not possible |
| | 04 | Execution not possible yet |
| | 05 | Will execute at ;<timestamp> |
| CmdID | HexString | |
| IP/Host | ASCII | IP or hostname of the platform |
| Media | BASE64 | List of video stream or image URLs |
| Text | BASE64 | Human readable free text description of the status |

Sample 1: STATUS;15;661D44C0;75DA;U;;;4;2;20;;50;;Fully operational

Sample 2: STATUS;16;661D64C0;129E;R;;;2;2;10;;;03;0F78;aHR0cDovLzEwLjAuMC4xL2ltYWdlLnBuZw==;Out of fuel!

2.9. ACKNOWLEDGE

Description: If a client or the SEC requested an acknowledge of a packet one has to use this this message. The acknowledgement flag is fixed set to FALSE. The awaiting client or SEC have to wait maximal one seconds before resending the original message with set acknowledgement flag.

Structure:
ACKNOWLEDGE;<Number>;<Time>;<Sender>;<Classification>;<MAC>;
<Recipient>(M);<Name of the message>(M);<Number of the message>(M)

| | | |
|-----------|-----------|---|
| Recipient | HexString | In most cases, you should use a hexadecimal string representation of a 16-bit unsigned integer, but you can also use freely selected textual identifiers. |
| Name | ASCII | The name of the message which should be acknowledged. |
| Number | Number | The number of the message which should be acknowledged. This is a hexadecimal string representation of an 8-bit. |

Sample: ACKNOWLEDGE;18;661D64C0;129E;R;;;FE2A;COMMAND;31

2.10. RESEND

Description: Missing messages can be requested again with this message. These messages can be recognized by the message number in the header, the sender ID and the message name as described in chapter IV.1.1.1.

Structure:

RESEND;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<Recipient>(M);<Name of the missing message>(M);<Number of the missing message>(M)

| | | |
|-----------|-----------|---|
| Recipient | HexString | In most cases, you should use a hexadecimal string representation of a 16-bit unsigned integer, but you can also use freely selected textual identifiers. |
| Name | ASCII | The name of the message which should resend. |
| Number | Number | The number of the message which should resend. This is a hexadecimal string representation of an 8-bit integer; |

Sample: RESEND;20;661D64C0;129E;R;;;FE2A;TEXT;31

2.11. GENERIC

Description: This message is an empty container for transporting any kind of data. It has to be defined in the respective case. For example, one can use it to exchange the original MESE/SEDAP messages or other propriety protocol data. In the last case you have to use any other self-defined type. If the coding indicator is not set, no coding is assumed.

Structure:

GENERIC;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<ContentType>;<Encoding>;<Content>

| | | |
|-------------|--------|--|
| ContentType | SEDAP | Content is an original MESE message |
| | ASCII | Self-defined ASCII string |
| | BINARY | Self-defined binary array |
| Encoding | BASE64 | Content is Base64 encoded |
| | NONE | Content is NOT encoded |
| Content | | Any content in printable ASCII or Base64 encoded |

Sample 1: GENERIC;5E;661D4410;66A3;R;;;SEDAP;FALSE;
Sample 2: GENERIC;5E;661D4410;66A3;R;TRUE;;SEDAP;TRUE;U2FtcGxIGJpbmFyeSBkYXRhIEdyZWV0aW5ncyA6RA==
Sample 3: GENERIC;5E;661D4410;66A3;R;;;RADNMEA;;\$RATTM,11,11.4,13.6,T,7.0,20.0,T,0.0,0.0,N,,Q,,154125.82,A,*17

2.12. HEARTBEAT

Description: This message offers the possibility to check the connection, which is primarily important, if you are using UDP or serial connection. It should not be sent more often than 1Hz. Nevertheless, if it is needed – one can use a faster repetition. The receiver field is optional and can be one single recipient or a list of more than one recipient. If no recipient is provided than all possible receivers in the network/serial net are addressed. A heartbeat message has an empty acknowledgement flag, cause you cannot request one for it. Besides this, the acknowledgement flag is fixed set to FALSE (empty field).

Structure:

HEARTBEAT;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;<Recipient>

Recipient HexString In most cases one should use a hexadecimal string representation of a 16-bit unsigned integer
but one can also use freely chosen textual identifier.

Sample 1: HEARTBEAT;42;661D5420;89AD;U;;;FE2A

Sample 2: HEARTBEAT;43;;1022

Sample 3: HEARTBEAT;43;

Sample 4: HEARTBEAT

2.13. KEYEXCHANGE

Description: If you don't have the possibility to exchange a password/key on another channel (e.g., mail, telco), this message can be used to exchange keys via Diffie-Hellman-Merkle or via the post-quantum Kyber process. It's preferred to use ECDH or standard DH with MAC DRBG. If possible, also use MAC authentication for these messages or otherwise do some plausibility checks. The current phase defines if a field is mandatory. Both sides can restart the process by simply sending a message with the phase set again to zero or 1 again. The other side should restart the whole process inclusive generating new key pairs if using ECDH or Kyber. Please pay attention, that the ECDH and Kyber always start with phase 1.

Key-Exchange-Process:

| | | | |
|---------------|----|---------|---|
| <i>DH:</i> | 1. | Phase 0 | A sending p, g (could also send public key but this should be done in the next phase 1 at first) |
| | 2 | Phase 1 | A & B sending public key |
| | 3. | Phase 3 | A & B generating shared secret key for AES encryption and sending confirmation of shared secret |
| <i>ECDH:</i> | 1. | Phase 1 | A & B sending public key |
| | 2. | Phase 3 | A & B generating shared secret key for AES encryption and sending confirmation |
| <i>Kyber:</i> | 1. | Phase 1 | A generates key pair and send public key to B |
| | 2. | Phase 2 | B generates shared secret for AES encryption, encrypts it with the public key of A and sending it |
| | 3. | Phase 3 | A sends confirmation |

Structure:

KEYEXCHANGE;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<Recipient>;<Phase>;<KeyLength>;<Prime>;<Natural Number>;<Public key>;<Encrypted secret key>

| | | |
|--------------------|-----------|---|
| Algorithm | 0 | DH (Diffie-Hellman-Merkle, NIST SP 800-56A/B, NIST SP 800-90A/B) |
| | 1 | ECDH (Diffie-Hellman-Merkle with Curve25519 / X25519, RFC 7748) |
| | 2 | Chrystals-Kyber (Post-Quantum Key-Encapsulation Mechanism) |
| Phase | 0 | Exchange the public variables and public key (DH only) |
| | 1 | Exchange public key(s) (ECDH and Kyber) |
| | 2 | Shared key successfully generated (MAC should also already generated with that key) |
| Key length | 128/256 | Bit Length of the key (Phase 0) |
| Prime (p) | HexString | Publicly known prime number (> 3000 bits / 375 byte) (Phase 0, DH only) |
| Natural number (g) | HexString | Publicly known natural number smaller than p (Phase 0, DH only) |
| Public key | BASE64 | The public key of the sender (Phase 1) |

Sample 1: KEYEXCHANGE;0;661D5420;89AD;U;;;FE2A;0;128;7FFFFFFF;822460DE

Sample 2: KEYEXCHANGE;0;661D5430;FE2A;U;;;89AD;1;128;;;6E6026EFF9D9EBEB9D4A973CB5C287DBD77D75EDDD2

3. SEDAP-Express JSON-Schema

The JSON schema was intentionally kept very simple. The JSON message contains only a list of one or more SEDAP-Express messages in their original (JSON-compatible) format. This means that the same message classes could be used for parsing and generating.

Schema:

```
{
  "messages":[
    {
      "message":""
    }
  ]
}
```

Sample:

```
{
  "messages":[
    {
      "message":"CONTACT;60;661B7410;66A3;S;TRUE;102;TRUE;53.32;8.11"
      "message":"METEO;AC;661D44C0;74BE;U;;15.4;15.5;;;10.2;72;20.3;;55;1005;25;;;2500;33"
      "message":"TEXT;D6;661D7032;324E;S;;3;"This is a chat message!";E4F1"
      "message":"GRAPHIC;79;661D62C0;910E;U;;8;1;FF8000;Area A;10000;53.43;9.45"
    }
  ]
}
```

4. SEDAP-Express Protobuf-Definition

The Google™ Protocol buffer definition is kept very simple, too. As with JSON messages, the same classes could be used to parse or generate the actual content.

Definiton:

syntax = "proto3";

```
message SomeMessage {
```

```
    message Messages {  
        string message = 1;  
    }
```

```
    repeated Messages messages = 1;  
}
```

V. Contact

Federal Armed Forces of Germany
Naval Support Command II A
c/o Volker Voß
Wibbelhofstraße 3
26384 Wilhelmshaven
Germany

Internet: [linkedin.com/in/volker-voss](https://www.linkedin.com/in/volker-voss)

E-Mail: mese@bundeswehr.org

Tel.: +49 4421 68 67290