Federal Armed Forces of Germany
UNIITY-Team

Classification
public or comparable
(Releasable to the internet)

Version 1.0
22.10.2025

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 UNIITY. 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 UNIITY 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

UNIITY	=	**U**nified, **N**etworked **I**ntegration of **I**nnovative **T**echnolog**Y**
SEDAP	=	Safety critical Environment for Data exchange And Process scheduling
CSV	=	Comma-Separated-Values
SEC	=	SEDAP-Express-Connector (Part of UNIITY)
MockUp/TestTool	=	Simulation of the real SEC and a C2-like system including a simple log, map and message creator
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 guidelines

1. Common conventions

- Basic format is CSV using ';' (0x3B) as separation character between two data fields, excess semicolons can be truncated
- Elements of list values are separated by '#' (0x23), List values are marked with a '*' in the specification
- Message ends with a '\n' (0x0A) as termination
- Mandatory fields/elements except the name are marked with (M)
- Maximum length of untyped or text fields is 256 bytes
- 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
- Support for IPv4 and 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.2.14). 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:
  - 32Bit/256Bit HMAC in combination SHA256 (FIPS 198-1 / RFC 2104)
  - 32Bit/128Bit CMAC in combination AES128 (NIST SP 800-38B)
  - 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;0195236D151A;66A3;R;;089A01E7;53.32;8.11;0;5.5;21;22;;;FGS Bayern

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:
  - AES128/256 CFB/NoPadding (NIST SP 800-38A)
  - AES128/256 CTR/NoPadding (NIST SP 800-38A)
  - AES128/256 ECB/PKCS7Padding (NIST SP 800-38A)
  - 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;01952384BD8D;66A3;R;;;53.32;8.11;0;5.5;21;22;;;FGS Bayern;sfspclff-------

Sample encrypted message (AES128, CFB, Password:expressexpressex):
UdlsDIB4oeKiuU4PXtV9qCPrrk10yPFg38Bakm7oGVOOC3siuczsyg37+Q9eiDE1Z0qyaXQl4puRGdB0mpb1Vf6cfhj7n7270Gv/VjW5Ol8IAFJh

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 due to the BASE64 encoding)
- 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 don't match a message name, prove for compression

Sample reference message:
TEXT;D3;01952381E21B;324E;S;TRUE;;;;1;NONE;"This is an alert!"

Sample compressed message:
C3GNCLF2MbY2MLQ0NTK2MHQ1MnSyNjYycbUOtg4JCnW1BgJDaz9/P1drpZCMzGIFIErMU0jMSS0qUVQCAA==

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, multiple messages per UDP-packet possible
- Standard Multicast-Address is 228.2.19.80 resp. ff02:8:2:19:80::1
- HEARTBEAT messages should be answered with UDP-Unicast if possible (see chapter IV.2.12)
- Is highly recommended to use the <number> and <acknowledgement> field in the header cause of possible packet lost

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:
    - Message never contains a \n (0x0A), therefore the message could be sent with an additional appended \n\n\n\n
    - Message contains one or more \n (0x0A), have to be BASE64-encoded and sent with an additional appended \n\n\n\n

8. MQTT connection

- Connection can establish via normal TCP/MQTT or via SSL/Certificates protocol
- Messages have to be published under "UNIITY-X/<senderid>/<messagetype>

9. REST-API

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

10. Protocol Buffers

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

IV. Specifications of the data exchange

1. General specifications

SEDAP-Express offers different kinds of connections. SEDAP Express offers various connection types. You can freely choose which one best suits your needs, or which one you have the most experience with or existing, adaptable code for.

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 a 7-bit sequential number. Each type of message has its own counter that starts again with zero after reaching 127/7F. A reconnect does not resets the counters! The limitation to 7-bit or a max of 127 prevents problems with signed/unsigned byte formats.
<Time>	A hexadecimal string representation of a 64-bit Unix time stamp with milliseconds.
<Sender>	You can use freely selected textual identifiers, e.g. “OKRA”. This field won’t be changed, even if a message has been forwarded or relayed. This sender identification can be chosen 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. For instance, if central unit relays information of drones in a swarm.
<Classification>	Classification level of the content: P=public, U=unclassified, R=restricted, C=confidential, S=secret, T=top secret
<Acknowledgement>	TRUE=request an acknowledgement, FALSE/Nothing=No acknowledgement. An acknowledgement of a messages requires a specified message number (second field of this header)!
<MAC>	A 32/128/256-bit wide hash-based message authentication code for verification (see chapter III.2 for more)
<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 and sample requests/answers 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. Message specifications

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 $359.999\overline{9}$, 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 KEYECHANGE 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 UNIITY network. If there is more than one “own unit”, the “Sender” field must be filled in to distinguish between the different units. In exceptional cases, different names can also be used for differentiation. The corresponding option must then be explicitly enabled in the SEC.

Structure:
OWNUNIT;<Number>;<Time>;<Sender>(M?);<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;0191C643A8AF;DRONEONE;R;;;53.32;8.11;0;5.5;21;22;;;FGS Bayern;SFSPCLFF-------
Sample 2: OWNUNIT;11;0191C643A8AF;22AA;U;FALSE;4389F10D;77.88;-10.12;5577.0;33.44;55.66;1.1;-2.2;3.3;Ownunit;SFGPIB----H----

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 UNIITY network. You have to choose between Lat/Lon/Alt OR relative distance – one is mandatory. If you choose relative distance, than you also have to provide a OWNUNIT (Chapter 2.1) message – otherwise the position of the receiver will be used as reference point.

Structure:
CONTACT;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;
<ContactID>(M);<DeleteFlag>;<Latitude>[°](M);<Longitude>[°](M);<Altitude>[m];
<relative X-Distance>[m](M);< relative Y-Distance>[m](M);<relative Z-Distance>[m](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 id of the contact chosen by the sender of this message
DeleteFlag	TRUE	Contact has to be removed
	FALSE	Contact is current
Name	ASCII	Name of the contact, maximum length 64 bytes
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, could be written in lower or upper case
MMSI	MMSI	Maritime Mobile Service Identity
ICAO	ICAO	International Civil Aviation Organization
Image	BASE64	Image data (JPG, PNG, TIF) encoded in BASE64, preferred length up to 65000 bytes
Comment	BASE64	Free text to the contact, maximum length 2048 bytes

Sample 1: CONTACT;5E;0191C643A8AF;83C5;R;;;100;FALSE;53.32;8.11;0;;;;120;275;;;;;;;FGS Bayern;AR;SFSPCLFF-------;;;;VXNlIENIMjI=
Sample 2: CONTACT;5F;0191C643A8AF;83C5;U;;;101;;36.32;12.11;2000;;;;44;;;;;;;;Unknown;O;;221333201;;;UG9zcyBOZXRoZXJsYW5kcw==
Sample 3: CONTACT;60;0191C643A8AF;4371;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 or the bearing or visually recognizable objects.

Structure:
EMISSON;<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	ASCII	A positive identification unique number or free text id 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	Wobbulated
	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
	1D	LASER Weapon
	1E	VISUAL Object
SIDC	SIDC	Identification code
Comment	BASE64	Free text to the emission, maximum length 2048 bytes

Sample 1: EMISSION;5E;0195238E15AD;66A3;R;;;100;;53.32;8.11;0;;;;20;8725000#8735000;20000;3;0;2;6;5;10233;;SA-8
Sample 2: EMISSION;5F;0195238E25AD;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;0195238E25AD;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>;<Recipient>;<Type>;<Encoding>;<Text>(M)

Recipient	ASCII	You can use a freely selected textual identifiers, as explained in the table from chapter IV.1.1.
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, maximum length 2048 bytes
Reference	ASCII	Reference to a contact or emission

Sample 1: TEXT;D3;0195238E25AD;324E;S;TRUE;;;1;NONE;"This is an alert!";1000
Sample 2: TEXT;D4;0195238E25CC;324E;C;TRUE;;;2;NONE;"This is a warning!"
Sample 3: TEXT;D5;0195238E25EF;324E;R;;;;3;;"This is a notice!"
Sample 4: TEXT;D6;0195238E285B;324E;U;;;ORKA;4;BASE64;IlRoaXMgaXMgYSBjaGF0IG1lc3NhZ2UhIg==

2.6. GRAPHIC

Description: Define graphical plans likes polygons, squares or routes

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

GraphicType	00	Point: <Latitude>[°];<Longitude>[°];<Altitude>[m]
	01	Path: <Latitude>[°],<Longitude>[°],<Altitude>[m] # …
	02	Polygon: <Latitude>[°],<Longitude>[°],<Altitude>[m] # …
	03	Rectangle: <RotationAngle>[°];<Latitude1>[°],<Longitude1>[°],<Altitude1>[m]#<Latitude2>[°],<Longitude2>[°],<Altitude2>[m]
	04	Square: <Latitude>[°];<Longitude>[°];<Altitude>[m];Radius-X[m];Radius-Y[m]
	05	Circle: <Radius>[m];<Latitude>[°];<Longitude>[°];<Altitude>[m]
	06	Ellipse: <Radius-X>[m];<Radius-Y>[m];<CenterLatitude>[°];<CenterLongitude>[°];<CenterAltitude>[m]
	07	Block: <Latitude>[°];<Longitude>[°];<Altitude>[m];X-Radius [m];Y-Radius [m];Z-Radius [m]
	08	Sphere: <Latitude>[°];<Longitude>[°];<Altitude>[m];Radius[m]
	09	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 in Web notation 800000FF for a darker red
FillColor	RGBA	Color of the filling in Web notation 00FF0080 for translucent green
TextColor	RGBA	Color of the text in Web notation 32CD32FF for lime
Encoding	BASE64	Text is BASE64 encoded
	NONE	Text is not encoded
Annotation	ASCII	Text for an annotation to this graphic, maximum length 32 bytes

Sample 1: GRAPHIC;79;0195238E35AD;910E;U;;;8;1;FF8000;;;BASE64;QXJlYSBBbHBoYQ==;10000;53.43;9.45
Sample 2: GRAPHIC;78;0195238E45AD;910E;U;;;1;1;808080;;FFFF00;;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 camera/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. Unix times stamps have to be written as hexadecimal string. Altitude is equivalent to depth.

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

Recipient	ASCII	You can use a freely selected textual identifiers, as explained in the table from chapter IV.1.1.
CmdID	HexString	16-Bit hexadecimal identifier of the command (0000 means all last commands)
CmdFlag	00	Add command
	01	Replace (last) command
	02	Cancel (last) command
	03	Cancel all commands (Same as 02 if there it is not a command sequence)
CmdType	00	Power off: <Unix time stamp>;<Power on unix time stamp>(optional)
	01	Restart: <Unix time stamp>
	02	Standby: <Unix time stamp>;<Wakeup unix time stamp>(optional)
	03	Sync time: <IP/Hostname of a NTP server>
	04	Send status
	05	Move: <Latitude>[°];<Longitude>[°];<Altitude>[m];<Tolerance>[m];<Unix time stamp>
	06	Rotate: <HeadingAngle>[°];<RollAngle>[°];<PitchAngle>[°]
	07	Loiter: <CenterLatitude>[°];<CenterLongitude>[°];<Altitude>[m];<Radius>[m]
	08	Scan Area: <Latitude1>[°];<Longitude1>[°];<Latitude2>[°];<Longitude2>[°];<Altitude>[m];<Rotation>[°]
	09	Take photo: <Number of camera>;<Camera mode>
	0A	Make video: <Number of camera>;<Camera mode>;<Duration>
	0B	Live video stream: <Number of camera>;<ON|OFF>;<Camera mode> (depends of the camera type)
	0C	<EngagementCmd[start-engagement|hold-engagement|stop-engagement]>;<WeaponID>;<ContactID>
	EE	Sanitize system (e.g. in a case of emergency or drone hijacking)
	FF	Generic Action: <Kind of action> (Has to be defined individually, inclusive implementing a custom connector software for UNIITY)

Sample 1: COMMAND;55;0195238E25AD;5BCD;S;TRUE;4389F10D;ORKA;1111;01;0C;hold-engagement;bomb;1000
Sample 2: COMMAND;29;0195238E35AD;E4B3;C;TRUE;;Drone1;;FF;00;OPEN_BAY
Sample 3: COMMAND;29;0195238F55AD;E4B3;C;TRUE;;Drone2;0000;03   (Cancel all last commands)

Sample 4: A complex move to with loitering at the end. The drone must calculate the speed itself based on the timestamp of the last move.
COMMAND;29;0195238E25AD;E4B3;C;TRUE;;ORKA;0331;00;05;53.5143;8.1574;50;5
COMMAND;2A;0195238E25AD;E4B3;C;TRUE;;ORKA;0331;00;05;53.4897;8.1908;50;5
COMMAND;2B;0195238E25AD;E4B3;C;TRUE;;ORKA;0331;00;05;53.4694;8.2131;50;5
COMMAND;2C;0195238E25AD;E4B3;C;TRUE;;ORKA;0331;00;05;53.4397;8.2262;50;5;6644D570 (approx. 16 minutes later)
COMMAND;2D;0195238E25AD;E4B3;C;TRUE;;ORKA;0331;00;07;53.4397;8.2262;1000;200

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>*;<CmdID>;<CmdState>;<IP/Host>;<Media>*;<Text>

TecStatus	0	Off/Absent
	1	Initializing
	2	Degraded
	3	Operational
	4	Fault
OpsStatus	0	Not operational
	1	Degraded
	2	Operational
AmmunitionLevel	String#%	Relative remaining ammunition: <name of the weapon>#<ammunition level>#....
FuelLevel	String#%	Relative remaining fuel capacity: <name of the fuel tank>#<ammunition level>#....
BatterieLevel	String#%	Relative remaining batterie capacity: <name of the batterie>#<batterie level>#....
CmdID	HexString	ID of the related COMMAND message (16-bit)
CmdState	00	Undefined
	01	Executed successfully
	02	Partially executed successfully
	03	Executed not successfully
	04	Execution not possible (yet)
	05	Will execute at ;<timestamp>
IP/Host	BASE64	IP or hostname of the platform, maximum length 64 bytes
Media	BASE64	List of video stream or image URLs, maximum length 4096 bytes
Text	BASE64	Human readable free text description of the status

Sample 1: STATUS;15;0195238E25AD;75DA;U;;;4;2;MLG#20;;Akku1#50;;;10.0.0.132;;RnVsbHkgb3BlcmF0aW9uYWw=
Sample 2: STATUS;16;0195238E25AD;129E;R;;;2;2;BMG#10;;;ED32;03;;aHR0cDovLzEwLjAuMC4xL2ltYWdlLnBuZw==;T3V0IG9mIGZ1ZWwh

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);<Type of the message>(M);<Number of the message>(M)

Recipient	ASCII	You can use a freely selected textual identifiers, as explained in the table from chapter IV.1.1.
Type	ASCII	The type of the message which should be acknowledged (e.g. CONTACT, RESEND, …).
Number	HexString	The number of the message which should be acknowledged. This is a hexadecimal string representation of an 7-bit.

Sample: ACKNOWLEDGE;18;0195238E25AD;129E;R;;;LASSY;COMMAND;2B

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	ASCII	You can use a freely selected textual identifiers, as explained in the table from chapter IV.1.1.
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 a 7-bit integer.

Sample: RESEND;20;0195238E25AD;129E;R;;;FE2A;TEXT;5D

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 UNIITY/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 SEDAP message
	ASCII	Content is a custom-defined ASCII string
	NMEA	Content is a NMEA0183 string
	XML	Content is in an XML structure
	JSON	Content is JSON formatted
	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 data, maximum length 8192 bytes

Sample 1: GENERIC;5E;0195238E25AD;66A3;R;;;JSON;;{"object": {"x": "1","y": "2"},"string": "Hello World"}
Sample 2: GENERIC;5E;0195238E25AD;66A3;R;TRUE;;BINARY;BASE64;U2FtcGxlIGJpbmFyeSBkYXRhIEdyZWV0aW5ncyA6RA==
Sample 3: GENERIC;5E;0195238E25AD;66A3;R;;;NMEA;NONE;$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 recipient 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 recipients in the network/serial net are addressed. A heartbeat message has an empty acknowledgement flag, because 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	ASCII	You can use a freely selected textual identifiers, as explained in the table from chapter IV.1.1.

Sample 1: HEARTBEAT;42;0195238E25AD;89AD;U;;;ORKA
Sample 2: HEARTBEAT;43;;1022
Sample 3: HEARTBEAT;43;
Sample 4: HEARTBEAT

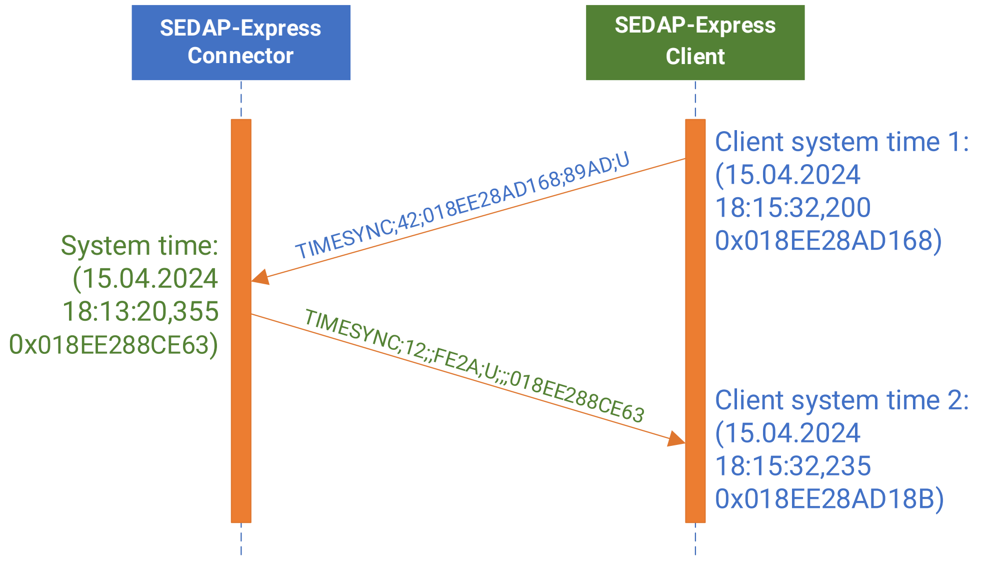
2.13. TIMESYNC

Description: A correctly synchronized time often plays an important role. In general, you should use operating system functions/command line tools and a dedicated time server. But if this is not available for reasons, there is this message dedicated for this purpose. If necessary, a client should perform time synchronization himself from time to time.
If a TIMESYNC message is received, the current system time of the recipient must be included as timestamp in the response. The original sender can then calculate the round-trip time (RTT) from half the difference between the time the response was received and the time the message was sent and add it to the transmitted timestamp. The result can then be adopted as the new system time.

Structure:
TIMESYNC;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;<Timestamp>

Timestamp	HexString	A hexadecimal string representation of a 64-bit Unix time stamp with milliseconds.

Sample 1: TIMESYNC;42;0191C643A8AF;89AD;U
Sample 2: TIMESYNC;12;;FE2A;U;;;0191C643A8AF


SEDAP-Express Connector
SEDAP-Express Client
Client system time 1: (15.04.2024 18:15:32,200 0x018EE28AD168)
TIMESYNC;42;018EE28AD168;89AD;U
System time: (15.04.2024 18:13:20,355 0x018EE288CE63)
TIMESYNC;12;;FE2A;U;;;018EE288CE63
Client system time 2: (15.04.2024 18:15:32,235 0x018EE28AD18B)

2.14. KEYEXCHANGE

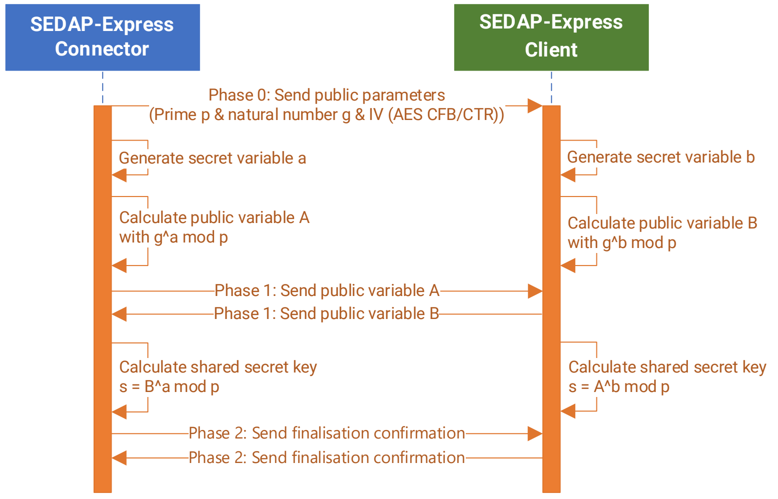
Description: If you don't have the possibility to exchange a password/key via another channel (e.g., mail, telco), this message can be used to exchange keys via Diffie-Hellman-Merkle or via the post-quantum Kyber-/FrodoKEM process. It's preferred to use ECDH or standard DH with MAC DRBG. If possible, 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 again. The other side should restart the whole process inclusive generating new key pairs if using ECDH or Kyber-/FrodoKEM. Please pay attention, that the ECDH has no phase 1. The whole process always ends by sending a finalization confirmation. To generate a 128- or 256-bit key from a larger secret key, "PBKDF2 with HMACSHA256" must be used with 1 iteration and no salt.

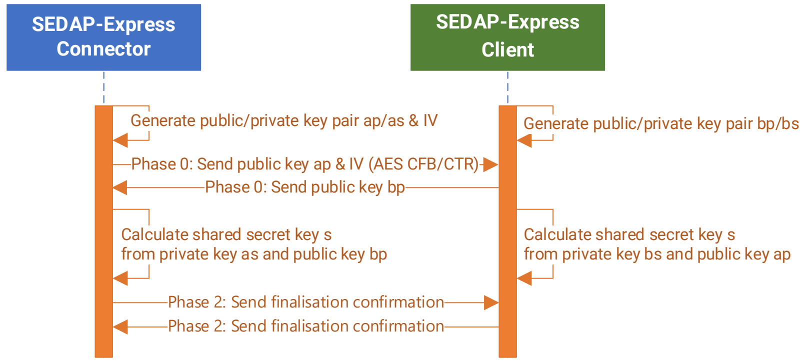
Structure:
KEYEXCHANGE;<Number>;<Time>;<Sender>;<Classification>;<Acknowledgement>;<MAC>;<Recipient>;<AlgorithmType>(M);<Phase>(M);<KeyLengthSharedSecret>(M);<KeyLengthKEM>(M);<Prime>(M);<Natural number>(M);<InitialisationVector>(M);<Public variable\key>(M)

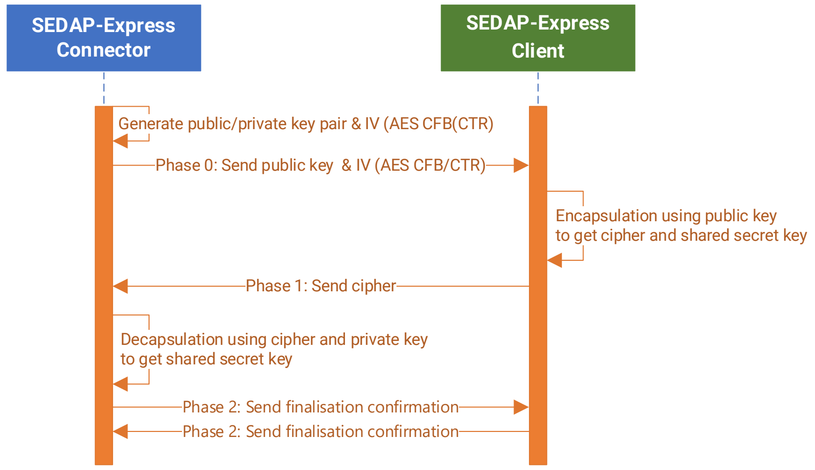
AlgorithmType	0	DH (Diffie-Hellman-Merkle 1024/2048 bit length, NIST SP 800-56A/B & 90A/B)
	1	ECDH (Diffie-Hellman-Merkle with Curve25519 / X25519, 1024/2048/4096 bit length)
	2	Kyber-512
	3	Kyber-768
	4	Kyber-1024
	5	FrodoKEM-640 (AES)
	6	FrodoKEM-976 (AES)
	7	FrodoKEM-1344 (AES)
Phase	0,1,2	Defines the current step in the key exchange process
KeyLengthSharedSecret	128, 256	Bit length of the shared secret key (Phase 0)
KeyLengthKEM	1024, 2048, 4096	Bit length of the key used for DH or KEM process (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)
Initialisation vector (IV)	HexString	Initialisation vector (IV) used for AES CFB/CTR encryption
Public variable/key	BASE64	The public variable/key of the sender

Sample 1: KEYEXCHANGE;0;0191C643A8AF;89AD;U;;;FE2A;0;128;1024;7FFFFFFF;822460DE
Sample 2: KEYEXCHANGE;0;0191C643A8AF;FE2A;U;;;89AD;1;128;2048;;6E6026EFF9D9EBEB9D4A973CB5C287DBD77D75EDDD2

DH-, ECDH and Kyber/FrodoKEM sequences:


SEDAP-Express Connector
SEDAP-Express Client
Phase 0: Send public parameters (Prime p & natural number g & IV (AES CFB/CTR))
Generate secret variable a
Generate secret variable b
Calculate public variable A with g^a mod p
Calculate public variable B with g^b mod p
Phase 1: Send public variable A
Phase 1: Send public variable B
Calculate shared secret key s = B^a mod p
Calculate shared secret key s = A^b mod p
Phase 2: Send finalisation confirmation
Phase 2: Send finalisation confirmation


SEDAP-Express Connector
SEDAP-Express Client
Generate public/private key pair ap/as & IV
Generate public/private key pair bp/bs
Phase 0: Send public key ap & IV (AES CFB/CTR)
Phase 0: Send public key bp
Calculate shared secret key s from private key as and public key bp
Calculate shared secret key s from private key bs and public key ap
Phase 2: Send finalisation confirmation
Phase 2: Send finalisation confirmation


SEDAP-Express Connector
SEDAP-Express Client
Generate public/private key pair & IV (AES CFB(CTR)
Phase 0: Send public key & IV (AES CFB/CTR)
Encapsulation using public key to get cipher and shared secret key
Phase 1: Send cipher
Decapsulation using cipher and private key to get shared secret key
Phase 2: Send finalisation confirmation
Phase 2: Send finalisation confirmation

:

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. With GET request the client can retrieve messages. On the other side with POST the client can send messages to the server. For simplicity there is only one endpoint at all. You can mix different messages in a single POST request. Updates for existing messages, e.g. contacts, can be sent to the server with POST or alternatively with PUT.

Endpoint:
http://<ip>:<port>/SEDAPEXPRESS

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

Sample GET request:

```
GET /SEDAPEXPRESS HTTP/1.1
Host: sample.host
Accept: application/json
```

Sample GET answer:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 380
{
  "messages":[
    {
      "message":"CONTACT;60;0191C643A8AF;66A3;S;TRUE;102;TRUE;53.32;8.11",
      "message":"METEO;AC;0191C643A8AF;74BE;U;;15.4;15.5;;;10.2;72;20.3;;55;1005;25;;;2500;33",
      "message":"TEXT;D6;0191C643A8AF;324E;S;;3;This is a chat message!;E4F1",
      "message":"GRAPHIC;79;0191C643A8AF;910E;U;;8;1;FF8000;Area A;10000;53.43;9.45"
    }
  ]
}
```

Sample POST request:

```
POST /SEDAPEXPRESS HTTP/1.1
Host: sample.host
Accept: application/json

{
  "messages":[
    {
      "message":" OWNUNIT;5E;0191C643A8AF;66A3;R;TRUE;;42.32;-123.11;10000;50.23;297;;;33.3;-0.15;sfapmf---------"
      "message":"TEXT;AE;0191C643A8AF;374E;S;;3;This is a chat message!;E4F1"
    }
  ]
}
```

Sample POST answer:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"success":"true"}
```

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

Please report any issues, suggestions, additions or whatever you have in mind with SEDAP-Express via GitHub, by email or by phone.

Federal Armed Forces of Germany
German Naval Support Command
UNIITY-Team c/o Volker Voß
Wibbelhofstraße 3
26384 Wilhelmshaven
Germany

GitHub:	github.com/UNIITY-Core/SEDAP-Express
Internet:	linkedin.com/in/volker-voss
E-Mail:	uniity@bundeswehr.org
Tel.:	+49 4421 68 67290