Simulator Communication System

Driven by CAN Aerospace

www.canaerospace.net

SCS relies on CAN Aerospace version 1.7 provided by Stock Flightsystems. While the CAN Aerospace protocol was developed for the CAN bus originally, SCS takes it into a UDP network. This specification describes the differences between the CAN and network based implementation. It is to be understood as an addendum to the original work of Stock Flightsystems and should be read after the original specification.

1 CAN Aerospace on CAN 2.0B (29 bit identifiers)

The CAN Aerospace distribution used by SCS is based on CAN 2.0B and thus uses both 11 bit standard and 29 bit extended identifiers. All communication participants must be able to read and write CAN messages with either 11 bit or 29 bit identifiers. Because the underlying CAN Aerospace 1.7 specification is based on 11 bit identifiers, SCS' identifier distribution follows the convention that all identifiers that are inherited directly from CAN Aerospace 1.7 must be in 11 bit format, while data offered exclusively by SCS and incompatible to CAN Aerospace 1.7 must be in 29 bit format.

2 ID Ranges

The following ID ranges are inherited from the CAN Aerospace 1.7 specification. SCS will only accept them if they are in 11 bit format:

- Emergency Event Data Channel is not used
- High Priority Node Service Data Channel (NSH): 128 - 199
- High Priority User-Defined Data Channel is not used
- Normal Operation Data Channel (NOD): 300 - 1799
- Low Priority User-Defined Data Channel is not used
- Debug Service Data Channel is not used
- Low Priority Node Service Data Channel is not used

The ids between 2032 and 2048 are unusable due to limitations of CAN 2.0A.

For the extended identifiers above 2048, SCS occupies the following ranges:

Message Type	CAN-ID Range	Explanation
SCS High Priority Node Service Data (SNSH)	128 – 129 ($80 - $81) but in 29 bit format	Operational commands specific to Simulator Communication System and not covered by CAN Aerospace 1.7
SCS Normal Operation Data (SNOD)	2300 – 3000 ($8FC - $BB8) in 29 bit format	Operational and status data specific to Simulator Communication System and not covered by CAN Aerospace 1.7

3 Data Types

The following user-defined data types are used:

Data Type	Range	Bits	Explanation	Type #
ACHAR5	0 to 255	5 x 8	ACHAR5 conveniently transmits ICAO waypoint identifiers in one CAN Aerospace message. 4 ASCII character are carried as in ACHAR4, and an additional ASCII character is carried in the service code byte, which is unused in NOD range. This is the last (5^{th}) character of the string.	100 ($64)
IACHAR4	0 to 255	4 x 8	I(ndexed)ACHAR4 transmits strings of length 6 to 954 (255 x 4) characters. 4 x ASCII character are carried as in ACHAR4. The service code is used as an index. It defines the 4 character aligned position of the transmitted substring in a larger string.	101 ($65)
VFLOAT	1-bit sign 32-bit fraction 8-bit exponent	32	V(ector)FLOAT transmits arrays of floating point numbers. Single precision floating-point value according to IEE-754-1985 with index in service code.	102 ($66)
VLONG	-2147483647 - +2147483648	32	V(ector)LONG transmits arrays of integer longs. 2's complement integer with index in service code.	103 ($67)

4 Node Services

The SCS specification supports the following node services on node service channel 0:

Node Service	Service Code	Response Required	Action
IDS	0	Yes	Identification service with configuration information (see below)
NSS	1	No	Node synchronization service, syncs all nodes to system-time.
Not supported	2 – 4		
TIS	5	Yes	Transmission interval service. In SCS this service is superseded by TIS29 (see below)

Not supported	6		
STS	7	No	State transmission service. Transmit all CAN messages once.
Not supported	8 – 11		
MIS	12	Yes	Module information service is used to determine which SCS software modules are loaded (see below).
MCS	13	Yes	Module configuration service is used to switch on and off SCS modules that offer certain capabilities.
Not supported	14		

The following user-defined node services are supported on SCS SNSH-channel 0 (ID 2176 - 2177):

Node Service	Service Code	Response Required	Action
DRS	100	No	Data request service. Addressed node should start cyclic transmission of normal operation data of specified identifier.
TIS29	101	Yes	29 bit data transmission interval service. Changes the priority and hence transmission interval to one of the four levels specified. Note that this is the only way to change the transmission interval of extended identifier messages, since TIS as defined by CAN Aerospace only works with 11 bit identifiers.
NCS	102	Yes	Dynamic node id configuration service. Is used to dynamically assign node ids to software nodes in the network.

4.1 Identification Service (IDS)

SCS uses the following convention for the four response bytes in an IDS:

Message Data Byte	Data Field Description	Service Request	Service Response
0	Node-ID	<target node-ID or broadcast>	<node-ID>
1	Data Type	NODATA	UCHAR4
2	Service Code	0	0
3	Message Code	<Sending node id>	<as in request>
4-7	Message Data	n.a.	Byte 0: Hardware Revision: The version number of SCS Byte 1: Software Revision: The version number of the SCS configuration file. Byte 2: Identifier distribution: 100 for this SCS identifier distribution Byte 3: Header type: CAN Aerospace standard header (0)

4.2 Node Synchronization Service (NSS)

The NSS message data in the CAN Aerospace 1.7 is a 32 bit value, which could be interpreted as the UNIX variant of the epoch seconds. SCS implements a format compatible to both Microsoft Flight Simulator and X-Plane:

Message Data Byte	Data Field Description	Service Request
0	Node-ID	<node-ID>
1	Data Type	UCHAR4
2	Service Code	1
3	Message Code	0
4	Message Data	UTC hours 0-23
5	Message Data	UTC minutes 0-59
6	Message Data	UTC seconds 0-59
7	Message Data	UTC days since Jan 1st

4.3 Transmission Interval Service (TIS)

TIS is superseded in SCS by TIS29, since TIS can only address 11bit identifiers.

4.4 Module Information Service (MIS)

The module information service is used to ask SCS which modules are loaded and currently hooked to the flight loop. For all modules available and their numbering, refer to the variables and triggers overview in separate document.

Message Data Byte	Data Field Description	Service Request	Service Response
0	Node-ID	<node-ID>	<node-ID>
1	Data Type	NODATA	BLONG
2	Service Code	12	12
3	Message Code	<0>	<as in request>
4-7	Message Data	n.a.	The active modules as a bit-field of 32bit width.

4.5 Module Configuration Service (MCS)

The various modules that make up SCS' capabilities can be enabled or disabled according to the needs of clients. For a list of the modules, refer to the variables and triggers overview in separate document.

Note the usage of the message code as a sender identification. This is used to ensure that only nodes that comply with the SCS software revision (config file version) can get data provided by modules that are non-standard.

Message Data Byte	Data Field Description	Service Request	Service Response
0	Node-ID	<node-ID of targeted node>	<as in request>
1	Data Type	USHORT2	BLONG
2	Service Code	13	13
3	Message Code	<node-ID of sending node>	<as in request>
4-5	Message Data	The number of the module to enable or disable	
6-7	Message Data	1 if module should be enabled, 0 if it should be disabled	The active modules as a bit-field of 32bit width (as in MIS)

4.6 Data Request Service (DRS)

Request the cyclic transmission of data. DRS is normally used as a broadcast, and if any node can fulfill the request, this node should immediately start transmitting the requested data. No node service response is required, as the transmitted data forms the acknowledge. If no node can deliver the requested data, it is left to the requester to time out and handle the situation appropriately. In systems without redundancy it is considered a fatal error if more than one node can provide the requested data.

Note the usage of the message code as a sender identification. This is used to ensure that only nodes that comply with the SCS software revision (config file version) can get data from the extended identifier range.

Message Data Byte	Data Field Description	Service Request
0	Node-ID	<usually broadcast node-ID>
1	Data Type	ULONG
2	Service Code	100
3	Message Code	<node-ID of sending node>
4-7	Message Data	The identifier requested to be enlisted for transmission, 11 or 29 bit in a 32bit unsigned long, with most significant bit set for a 29bit identifier, not set otherwise

4.7 Transmission Interval Service 29 bit (TIS29)

For data enlisted for periodic transmission, the transmission interval can be set according to the

four levels defined by SCS. Note that the CAN Aerospace 1.7 TIS does not work with extended identifiers, because they do not fit in a 16 bit short integer. So for SCS this service is the only way of changing the transmission interval.

Note that this change in the transmission interval is not persistant in the SCS configuration, so a cold-start will always yield the standard transmission intervals as preset in the configuration.

Message Data Byte	Data Field Description	Service Request	Service Response
0	Node-ID	<node-ID>	<node-ID>
1	Data Type	ULONG	NODATA
2	Service Code	101	101
3	Message Code	255 = default as in config file 0 = high transmission rate 1 = middle transmission rate 2 = low transmission rate 3 = ultra low transmission rate	0 = OK -6 = CAN identifier or transmission rate out of range
4-7	Message Data	The identifier in question, 11 or 29 bit in a 32 bit unsigned long, with most significant bit set for a 29 bit identifier, not set otherwise.	n.a.

4.8 Node Configuration Service (NCS)

A typical simulator setup involves various hardware and software nodes, all communicating over the same CAN Aerospace bus, the software nodes attached via UDP. All nodes need a unique node id to perform node service requests or to sign their NOD. While hardware nodes are assigned their node-id typically in their firmware, the limited range of only 255 unique IDs renders giving every software component a hard-wired node-id impossible. Therefore each software that uses SCS is required to have a unique 32bit identifier. Using a properly seeded 32bit PRNG will yield a reasonably safe UID for all practical applications.

Each software node is required to set node id 255 (all bits set) on start up and try to acquire a node id for this session via NCS as follows:

Message Data Byte	Data Field Description	Service Request	Service Response
0	Node-ID	<broadcast node-ID 0>	<broadcast node-ID 0>
1	Data Type	ULONG	ULONG
2	Service Code	102	102
3	Message Code	0	The new node id assigned to this node, valid for this session.
4-7	Message Data	The 32bit unsigned integer unique id. (Usually generated with a PRNG)	The 32bit unsigned integer unique id.

If the node loses connection to SCS, which means SCS drops out of his trusted clients list, the node id must be reverted to 255 and a new request issued.

5 Identifier distribution

All data in the normal range and in 11 bit format can be requested by anyone. Data with 29 bit identifiers can only be requested by participants that passed an IDS compliance check and rely on the revision of the configuration file that SCS uses.

The 11 bit flight data offered by SCS are according to the CAN Aerospace 1.7 standard identifier distribution. Wherever the standard identifier distribution offers data to be send as either FLOAT or SHORT2, SCS uses FLOAT.

The 29 bit parameters offered by SCS are:

CAN identifier	Parameter Name	Data Type	Units	Notes
418	Throttle lever position	VFLOAT	Percent	Throttle lever #0-7 in service code
500	Engine N1	VFLOAT	Percent	Engine #0-7 in service code
504	Engine N2	VFLOAT	Percent	Engine #0-7 in service code
520	Engine EGT	VFLOAT	K	Engine #0-7 in service code
524	Engine Fuel Flow	VFLOAT	Kg/h	Engine #0-7 in service code
1100-1103	VHF#n COM frequency	LONG	kHz	LONG kHz instead of MHz float to eliminate round-off errors
1104-1107	VOR/ILS #n frequency	LONG	kHz	LONG kHz instead of MHz float to eliminate round-off errors
1500 - 1509 ($5DC - $5E5)	Electrical switches	BCHAR	On or Off	1500: Avionics power 1501: Main battery power
1510 - 1519 ($5E6 - $5EF)	Lighting switches	BCHAR	On or Off	1510: Beacon lights 1511: Strobe lights 1512: Landing lights 1513: Nav lights 1514: Taxi lights 1515: Panel backlight 1516: Panel floodlight
1520 - 1529 ($5F0 - $5F9)	Icing system switches	VFLOAT BCHAR	Ratio On or Off	1520: Engine anti-ice 1521: Pitot heat
1530 - 1549 ($5FA - $60D)	Autopilot states	VARIOUS	various	See custom data document for detailed description.

1550 - 1559 ($60E-$617)	Aircraft specific constants	FLOAT LONG LONG VFLOAT	Knots Mach # # kg	1550: V_{S0} 1551: V_S 1552: V_C 1553: V_{MD} 1554: M_{MO} 1555: number of engines 1556: number of flap detents 1557: total fuel capacity
1560 - 1569 ($618 - $621)	Secondary flight controls	VFLOAT VFLOAT BCAHR FLOAT FLOAT FLOAT FLOAT	Ratio ratio on or off degrees ratio ratio m/s	1560: gear deploy as array (front, left, right) 1561: thrust reverser deploy ratio 1562: speed brake armed 1563: flaps degrees 1564: flaps percent 1565: slats percent 1566: barber pole speed
1570 - 1579 ($622 - $62B)	Fuel and weight	FLOAT	Kg	1570: Total gross weight 1571: Total fuel on board weight
1580 - 1589 ($62C - $635)	EFIS selectors	LONG	Three positions	1580: Captains ADF1/VOR1 selector 1581: Captains ADF2/VOR2 selector
1590 – 1599 ($636 - $63F)	Annunciation lights	BCHAR	bool	1590: cabin door open 1591: fasten seat belt sign 1592: no smoking sign
1600 - 1649 ($640 - $671)	Tuned nav station data	VARIOUS	various	See custom data document for detailed information.
1650 - 1659 ($672 - $67B)	VasFMC specific variables	BCHAR VFLOAT	On or off ratio	1652: Disconnect throttle input axis 1655: Overridden throttle input (per axis)
1660 - 1699 ($67C - $6A3)	Simulator specific variables	BCHAR FLOAT FLOAT FLOAT	On or Off hPa K m/s	1660: sim is paused 1661: environmental QNH 1662: dewpoint at sealevel 1663: speed of sound at current PALT
1700 - 1720 ($6A4 - $6B8)	MCP specific variables	VARIOUS	various	MCP driver: detailed description and pricing available on request

6 CAN Aerospace over UDP

6.1 Data format and structure

The various data types that can be transmitted in the message data are combined in this union:

```
union canAS_Data_t {
    float       flt;        //!< floating point number.
    uint32_t    uLong;      //!< Unsigned 32 bit integer.
    int32_t     sLong;      //!< Signed 32 bit integer.
    uint16_t    uShort[2];  //!< 2 x unsigned 16 bit integer.
    int16_t     sShort[2];  //!< 2 x signed 16 bit integer.
    uint8_t     uChar[4];   //!< 4 x unsigned 8 bit integer.
    int8_t      sChar[4];   //!< 4 x signed 8 bit integer.
    char        aChar[4];   //!< 4 x unsigned ASCII character.
};
```

The data bytes (CAN 2.0 payload) are stored in this C-struct:

```
struct canAS_t {
    uint8_t         nodeId;         //!< Id of transmitting/receiving node.
    uint8_t         dataType;       //!< Id for CAN Aerospace message data type.
    uint8_t         serviceCode;    //!< Service code
    uint8_t         messageCode;    //!< Message code
    canAS_Data_t    data;           //!< CAN Aerospace message data.
};
```

To retain compatibility with other protocols, the CAN payload itself may either be 8 arbitrary bytes or a CAN Aerospace message. This is ensured by this union:

```
union can_Data_t {
    uint8_t     byte[CAN2AB_PAYLOAD];   //!< Raw CAN message.
    canAS_t     aero;                   //!< CAN Aerospace message.
};
```

The whole CAN message is stored in this struct containing the ID as 32 bit unsigned integer with the highest bit set for 29 bit identifiers, the data itself, and the data length code.

```
struct can_t {
    uint32_t    id;     //!< CAN id, for both 11 and 29 bit. MSB high for id29
    can_Data_t  msg;    //!< CAN message data bytes.
    uint8_t     dlc;    //!< Data length code, number of data bytes.
};
```

Make sure to NOT use `#pragma pack` when defining this struct, so the compiler can align properly. The `sizeof(can_t)` must equal 16 bytes.

A UDP message may consist of 1 to 71 CAN messages before IP-fragmentation occurs. Normally a UDP packet should contain 1 CAN message. SCS groups NOD array values, i. e. all Ixxxx data types into a single UDP packet containing all values of the array. Engine, gear, flap and string data use this mechanism.

6.2 UDP multicast settings

Multicast group is 239.40.41.42, port is 50707, ttl is 2.