ANSI E1.31-2018 Protocol Compliance Checklist

Green = Passed, White = Not Checked, Purple = Outside Library Scope, Blue = Not Implemented/Attempted, Grey = Already covered by another point

Section of ANSI E1.31-2018	Specific functionality required for compliance, functionality listing ommitted or greyed out if already checked in a previous section.	Test(s) which show compliance
1.2 Overview and Architecture	- Allows transfer of arbitary START code DMX512-A data:	test_send_recv_single_universe_alternative_startcode_multicast_ipv4, test_send_recv_single_universe_multicast_ipv4, test_send_recv_single_universe_multicast_ipv6, test_send_recv_single_universe_alternative_startcode_multicast_ipv6
	- DMX data can be synchronized across multiple receivers using universe syncronisation:	test_send_across_universe_multiple_receivers_sync_multicast_ipv4, test_send_across_universe_multiple_receivers_sync_multicast_ipv6
	- Uses a ACN wrapper meaning it is compatiable with devices following the ANSI E.1.17 [ACN] standard:	test_sync_packet_transmit_format, test_discovery_packet_transmit_format, test_data_packet_transmit_format
	- Uses UDP as the transport/IP layer protocol:	test_data_packet_transmit_format, test_sync_packet_transmit_format
	- Supports multicast addressing:	test_send_recv_single_universe_multicast_ipv6, test_send_recv_single_universe_multicast_ipv4
	- Supports unicast addressing:	test_send_recv_single_universe_unicast_ipv6, test_send_recv_single_universe_unicast_ipv4
1.3 Appropriate Use of This Standard	- Uses UDP to provide a non-reliable IP transport mechanism:	
	- Allows multiple senders and receivers:	test_three_senders_three_recv_multicast_ipv4, test_three_senders_three_recv_multicast_ipv6
1.4 Classes of Data Appropriate for Transmission	- Allows transfer of arbitary START code DMX512-A data:	
1.5 Universe Synchronization	- Allows synchronisation through the universe synchronisation mechanism:	
1.6 Universe Discovery	- Allows universe discovery through the universe discovery mechanism:	test_universe_discovery_one_universe_one_source_ipv4, test_universe_discovery_one_universe_one_source_ipv6
3.5 Source	- A source is uniquely identified by the CID in the header of the packet:	Relies on library user to ensure CID's are unique, protocol doesn't specify a mechanism for this
	- A source may send multiple streams of data for different universes:	test_send_recv_two_universe_multicast_ipv4, test_send_recv_two_universe_multicast_ipv6
	- Multiple sources may output data for a given universe:	test_two_senders_one_recv_same_universe_no_sync_multicast_ipv4, test_two_senders_one_recv_same_universe_no_sync_multicast_ipv6
3.6 Receiver	- A receiever may listen on multiple universes:	test_two_senders_one_recv_same_universe_no_sync_multicast_ipv4, test_two_senders_one_recv_same_universe_no_sync_multicast_ipv6
3.7 Active Data Slots	- Sources for E1.31 should specify the location and amount of active data slots using the DMP First Property Address and DMP Property Count fields (shown in Table 4-1):	data_parse_tests::test_data_packet_parse_pack, data_parse_tests:: test_malformed_data_packet_dmp_layer_too_low_property_count_parse, data_parse_tests:: test_malformed_data_packet_dmp_layer_too_high_property_count_parse,
3.8 E1.31 Data Packet	- Identified by being transmitted with the VECTOR_E131_DATA_PACKET vector:	data_parse_tests::test_data_packet_parse_pack, data_parse_tests:: test_malformed_data_packet_extended_acn_vector_parse, data_parse_tests:: test_malformed_data_packet_dmp_layer_wrong_vector_parse
3.9 E.31 Synchronization Packet	- Contains only universe synchronisation information and no additional data:	sync_parse_tests::test_synchronization_packet_parse_pack
	- Identified by being transmitted with the VECTOR_E131_EXTENDED_SYNCHRONIZATION vector:	sync_parse_tests::test_synchronization_packet_parse_pack, sync_parse_tests:: test_sync_packet_framing_layer_unknown_vector, sync_parse_tests:: test_sync_packet_framing_layer_discovery_vector
3.10 E1.31 Universe Discovery Packet	- Identified by being transmitted with the VECTOR_E131_EXTENDED_DISCOVERY vector:	discovery_parse_tests::test_discovery_packet_parse_pack, discovery_parse_tests:: test_discovery_packet_unknown_framing_vector_parse, discovery_parse_tests:: test_discovery_packet_sync_framing_vector_parse
4 Protocol Packet Structure Summary	- E1.31 components must support the £1.31 Data Packet and E1.31. Universe Discovery Packet:	test_send_recv_across_universe_multicast_ipv4, test_send_recv_across_universe_multicast_ipv6, test_universe_discovery_one_universe_one_source_ipv4, test_universe_discovery_one_universe_one_source_ipv6
	- E1.31 components may support the E1.31 synchronization packet:	
4.1 E1.31 Data Packet	- Data is formatted as specified in Table 4-1	data_parse_tests::test_data_packet_parse_pack
	- Detection of malformed packets:	data_parse_tests
	- All packet content must be transmitted in network byte order (big endian):	data_parse_tests
4.2 E1.31 Synchronization Packet	- A universe can be used as a synchronisation universe and to transmit data on simultaneously:	test_send_recv_across_universe_multicast_ipv4, test_send_recv_across_universe_multicast_ipv6
	- Packet is formatted as specified in Table 4-2	sync_parse_tests::test_synchronization_packet_parse_pack
	- Detection of malformed packets:	sync_parse_tests
	- All packet content must be transmitted in network byte order (big endian):	sync_parse_tests
4.3 E1.31 Universe Discovery Packet	- A set of universe discovery packets shall be sent once every E131_UNIVERSE_DISCOVERY_INTERVAL:	test_universe_discovery_interval_ipv4
	- The list of E1.31 universes must be sorted:	test_discovery_packet_random_order_parse
	- The list of universes may include synchronisation universes:	test_universe_discovery_multiple_universe_one_source_ipv4
	- If the list of universes changes within an E131_UNIVERSE_DISCOVERY_INTERVAL a source may send upto one additional set of packets to update the information:	Source only sends updates on interval only: test_universe_discovery_interval_with_updates_ipv4
	- Packet is formatted as specified in Table 4-3	discovery_parse_tests::test_discovery_packet_parse_pack,
	- Detection of malformed packets:	discovery_parse_tests
	- All packet content must be transmitted in network byte order (big endian):	discovery_parse_tests

5 E1.31 use of the ACN Root Layer Protocol	- All E1.31 packets should use the ACN Root Layer Protocol as defined in ANSI E1.17 [ACN] specifically the fields specified in Table 5-4 which is for E1.31 on UDP.	data_parse_tests::test_data_packet_parse_pack, sync_parse_tests:: test_synchronization_packet_parse_pack, discovery_parse_tests:: test_discovery_packet_parse_pack,
	- Detection of malformed packets:	
5.1 Preamble Size	- The preamble size field must be 0x0010:	data_parse_tests::test_data_packet_parse_pack
	- Packets with a different preamble size must be discarded:	data_parse_tests::test_malformed_data_packet_wrong_preample_upper_byte_parse, data_parse_tests::test_malformed_data_packet_wrong_preample_lower_byte_parse
	- The preamble (preamble size field, post-amble size field and ACN packet identifier) length must match the size given in the field (0x10 octets):	data_parse_tests::test_data_packet_parse_pack data_parse_tests::test_malformed_data_packet_wrong_preample_upper_byte_parse, data_parse_tests::test_malformed_data_packet_wrong_preample_lower_byte_parse
5.2 Post-amble Size	- There is no post amble for RLP over UDP so the post-amble size field must be 0 and E1.31 receivers must discard packets if the post-amble size is not 0x0000.	data_parse_tests::test_malformed_data_packet_wrong_postample_upper_byte_parse, data_parse_tests::test_malformed_data_packet_wrong_postample_lower_byte_parse
5.3 ACN Packet Identifier	- The ACN packet identifier must be exactly 0x41 0x53 0x43 0x2d 0x45 0x31 0x2e 0x31 0x37 0x00 0x00 0x00 and must discard packets if the ACN packet identifier doesn't match above:	data_parse_tests::test_malformed_data_packet_wrong_acn_identifier_parse, data_parse_tests:: test_data_packet_parse_pack
5.4 Flags & Length	- The PDU length must be encoded in the low 12 bits of the root layer flags and length field:	test_malformed_data_packet_root_layer_too_low_length, test_malformed_data_packet_root_layer_too_high_length
	- The flags (top 4 bits) must be 0x7:	test_malformed_data_packet_root_layer_wrong_flags
	- The PDU length is computed started with octet 16 and counting all remaining octets in the packet including all payload:	test_malformed_data_packet_root_layer_too_low_length, test_malformed_data_packet_root_layer_too_high_length
	- A ful payload data packet should have a length of 638 octets:	test_data_packet_full_length_expected
	- A synchronisation packet should have a length of 49 octets:	test_sync_packet_length
	- A universe discovery packet length should be computed to the end of the list of universes field:	discovery_parse_tests
5.5 Vector	The root layer vector must be VECTOR_ROOT_E131_DATA if the packet contains E1.31 data:	test_data_packet_parse_pack, test_malformed_data_packet_unknown_acn_vector_parse, test_malformed_data_packet_extended_acn_vector_parse
	The root layer vector must be VECTOR_ROOT_E131_EXTENDED if the packet is for universe discovery or synchronisation:	test_synchronization_packet_parse_pack, test_sync_packet_root_layer_data_vector_parse, test_sync_packet_root_layer_unknown_vector_parse, test_discovery_packet_parse_pack, test_discovery_packet_root_layer_unknown_vector_parse, test_discovery_packet_root_layer_data_vector_parse
	Receivers must discard a packet if the vector isn't one of the above	
5.6 CID (Component Identifier)	Must be a UUID - a universally unique identifier that is 128 bit number unique across space and time:	Provided by the user of the library, not the responsibility of the library
	The CID must be compliant with RFC 4122 [UUID]:	Provided by the user of the library, not the responsibility of the library
	A piece of equipment must maintain the same CID for its entire lifetime:	Provided by the user of the library, not the responsibility of the library
	Must be transmitted in network byte order (big endian):	Provided by the user of the library, not the responsibility of the library
6.1 Flags & Length	- Each framing layer must start with the flags & length field, The field must be 16 bit with the PDU length encoded in the low 12 bits and 0x7 in the top 4 bits, The PDU length must be computed starting with octet 38 and continue through the last octet provided by the underlying layer	test_malformed_data_packet_framing_layer_wrong_flags_parse, test_malformed_data_packet_framing_layer_low_length_parse, test_malformed_data_packet_framing_layer_high_length_parse, test_sync_packet_framing_layer_wrong_flags_parse, test_sync_packet_framing_layer_length_too_long_parse, test_sync_packet_framing_layer_length_too_short_parse, test_discovery_packet_framing_layer_wrong_flags_parse, test_discovery_packet_framing_layer_length_too_short_parse, test_discovery_packet_framing_layer_length_too_long_parse
	- An E1.31 Data Packet with full payload must have a length of 638:	
	- An E1.31 Universe Discovery Packet must have a length between 120 and 1144 depending on the list of universes:	test_discovery_packet_no_universes, test_discovery_packet_max_universe_capacity
6.2 E1.31 Data Packet Framing Layer	- The packet must be formatted as specified in Table 6-5:	data_parse_tests
6.2.1 E1.31 Data Packet: Vector	- The E1.31 layer vector must be VECTOR_E131_DATA_PACKET for an E1.31 Data Packet	test_data_packet_parse_pack, test_malformed_data_packet_framing_layer_wrong_vector_parse
6.2.2 E1.31 Data Packet: Source Name	- The source name must be null-terminated:	test_malformed_data_packet_source_name_not_null_terminated_parse, test_data_packet_max_source_name_length_parse
	- The source name of a component must match the UACN field as specified in EPI 19 [ACN]:	Left to the user as source name is provided
	- The source name may be the same across multiple universes sourced by the same component:	Left to the user as source name is provided
	- The source name should be unique:	Left to the implementer / user-configuration as not specified in protocol how this should be done
6.2.3 E1.31 Data Packet: Priority	- The most recent E1.31 Data Packet from a single source must supersede any previous packet from that source:	This refers to how the data is treated on the device after the implementation since the implementation parses the data and returns it to the user immediately (no background parsing). The only time data waits is when waiting for synchronisation in which case the highest priority packet is kept and if at the same priority then the latest packet. test_store_2_same_universe_diff_priority_waiting_data, test_store_2_same_universe_same_priority_waiting_data, test_send_recv_diff_priority_same_universe_multicast_ipv4, test_send_recv_two_packets_same_priority_same_universe_multicast_ipv4
	- Data from sources with a higher priority (e.g. 200 vs 100) will be treated as the defininive data for that universe.	
	- If the E1.31 receiver is also doing universe syncronisation then the behaviour is undefined:	
	- A receiver may receive data for the same universe from multiple sources which is distinguished by examining the CID in the packet:	test_two_senders_one_recv_same_universe_no_sync_multicast_ipv4, test_two_senders_one_recv_same_universe_no_sync_multicast_ipv6

	- The priority field must be in the range 0 to 200	test_data_packet_lowest_priority_parse, test_malformed_data_packet_too_high_priority_parse, test_send_above_priorty
6.2.3.1 Multiple Sources at Highest Priority	- If there are multiple sources transmitting data at the same highest currently active priority for a given universe then this must be handled:	test_two_senders_one_recv_same_universe_custom_merge_fn_sync_multicast_ipv4
	- If a receiver is only capable of processing a certain number of sources of data it will encounter a sources exceeded condition when a greater number of sources are present:	Left to implementer to decided the number of sources allowed, provided as an argument when creating receiver. test_receiver_sources_exceeded_3, test_receiver_source_limit_2, test_receiver_source_limit_2_termination_check
6.2.3.2 Note on Merge and Arbitration Algorithms	- Allow various merging algorithms for combining data from multiple sources:	User can provide an alternative merge function for the part within the implementation (during synchronisation) test_two_senders_one_recv_same_universe_custom_merge_fn_sync_multicast_ipv4
6.2.3.3 Note on Resolution of Sources Exceeded Condition	- Various possible resolution mechanisms should be possible:	Left to the implementer, dependent on computational resources avaliable, not limited by library
	- Resolution mechanisms are recommended to not generate different results from the same source combination on different occasions as it can make troubleshooting more difficult:	
6.2.3.4 Requirements for Merging and Arbitrating	- The ability to merge/arbitrate between multiple sources, the maximum number of sources which can be handled and the algorithm used should all be declared in user documentation for the device:	
6.2.3.5 Requirements for Sources Exceeded Resolution	- The resolution behaviour for equipment to resolve a source exceeded condition should be specified in the user documentation:	
	- The sources exceeded condition is highly recommended to be easily detected at the device aswell as potentially through the network:	Left to the implementer, dependent on specific device
6.2.3.6 Requirements for Devices with Multiple Operating Modes	- All different operating modes for a device should be compliant with the standard or or non-compliant configurations should be clearly declared as such.	This library aims to be compliant however the device might have other modes
6.2.4.1 Synchronization Address Usage in an E1.31 Data Packet	- A synchronisation address of value 0 indicates that the data isn't synchronised meaning any waiting data must be discarded and the data acted on immediately.	test_send_recv_sync_then_nosync_packet_same_universe_multicast_ipv4
	- A nonzero synchronization address means that the data is synchronised, if the receiever doesn't support universe synchronisation the packet should be processed normally:	Doesn't apply as the implementation supports universe synchronisation.
	- A nonzero synchronisation address means that the data packet should be held until the arrival of the corresponding E1.31 synchronisation packet to release it:	test_send_across_universe_multiple_receivers_sync_multicast_ipv4, test_send_across_universe_multiple_receivers_sync_multicast_ipv6
	- A receiver must not synchronise any data until it has receieved its first E1.31 synchronisation packet on the synchronisation address:	
6.2.5 E1.31 Data Packet: Sequence Number	- Sources must maintain a sequence number for each universe transmitted:	
	- The sequence number should be incremented by one for each packet sent on the universe:	test_track_data_packet_seq_numbers
6.2.6 E1.31 Data Packet: Options	- The most significant bit is the Preview_Data, when set to 1 this means that the data is intended for use that doesn't affect the live output e.g. for visualisers or media server preview applications:	test_preview_data_2_receiver_1_sender
	- The Stream_Terminated bit (2nd most significant) triggers the termination of a stream or universe synchronisation without waiting for timeout and to indicate that the termination is not due to a fault condition. When set to 1 the source of data for the universe specified has terminated transmission of the universe:	test_termination_packet_empty_property_values_parse, test_termination_packet_partial_property_values_parse, test_termination_packet_full_property_values_parse
	- A source should send three packets when terminating the universe source:	test_terminate_stream
	- A receiver should enter network data loss condition when a packet with the stream terminated bit is set:	test_receiver_source_limit_2_termination_check
	- A receiver should ignore any property values in a packet with the stream termination bit set:	test_termination_packet_empty_property_values_parse, test_termination_packet_partial_property_values_parse, test_termination_packet_full_property_values_parse
	- The Force_Synchronisation bit (3rd most significant) says how a receiver should handle the loss of synchronisation, if set to 0 then on synchronisation loss the reciever must not update / process any new packets until syncronisation is re-established / resumes:	Not implemented as part of the project
	- If the Force_Synchronisation bit is set to 1 then if synchronisation is lost receivers may continue to process new E1.31 data packets without having to wait for synchronisation to resume / re-etablish:	Not implemented as part of the project
	- The least significant 5 bits of the field are reserved for future use and must be transmitted as 0:	test_data_packet_transmit_format
	- The least significant 5 bits of the field should be ignored by receivers:	test_data_packet_options_bit_4_set_parse, test_data_packet_options_bit_3_set_parse, test_data_packet_options_bit_2_set_parse, test_data_packet_options_bit_1_set_parse, test_data_packet_options_bit_0_set_parse
6.2.7 E1.31 Data Packet: Universe	- Universe values must be in the range 1 to 63999 inclusive, other universe values are reserved for future use and must not be used except for the E131_DISCOVERY UNIVERSE:	test_malformed_data_packet_too_low_universe_parse, test_malformed_data_packet_too_high_universe_parse, test_register_min_universe, test_register_max_universe, test_register_discovery_universe, test_register_above_max_universe, test_register_below_min_universe
	- The E131_DISCOVERY_UNIVERSE: is used for universe discovery:	test_discovery_packet_transmit_format
6.3 E1.31 Synchronization Packet Framing Layer	- The synchronisation packet framing layer must conform to Table 6-6:	sync_parse_tests

6.3.1 E1.31 Synchronization Packet: Vector	- The E1.31 layer vector must have a value of VECTOR_E131_EXTENDED_SYNCHRONIZATION for an E1.31 Synchronization Packet:	
6.3.2 E1.31 Synchronization Packet: Sequence Number	- Sources must maintain a sequence number for each universe transmitted:	
	- The sequence number should be incremented by one for each packet sent on the universe:	test_track_sync_packet_seq_numbers
6.3.3.1 Synchronization Address Usage in an E1.31 Synchronization Packet	- A synchronisation packet with a synchronisation address of 0 is meaningless as the entire purpose of the packet is to be used for universe synchronisation so should never be transmitted:	test_sync_addr_0
	- A synchronisation packet with a synchronisation address of 0 should be ignored by receievers:	test_sync_packet_too_low_sync_addr
	- When sending via multicast synchronisation packets must be sent only to the address corresponding to the synchronisation address:	test_sync_packet_multicast_address
	- Receievers may ignore synchronization packets sent to multicast address not corresponding to synchronisation addresses:	This implementation does not ignore packets sent to the wrong multicast universe. test_send_recv_wrong_multicast_universe
6.3.4 E1.31 Synchronization Packet: Reserved	- Octets 47-48 of a E1.31 Synchronisation packet are reserved for future used and must be transmitted as 0:	test_sync_packet_transmit_format
	- Octets 47-48 of a E1.31 Synchronisation packet must be ignored by receievers:	test_sync_packet_arbitary_reserved
6.4 E1.31 Universe Discovery Packet Framing Layer	- Packets must be formatted as specified in Table 6-7:	discovery_parse_tests
6.4.1 E1.31 Universe Discovery Packet: Vector	- E1.31 Universe Discovery Packets must have the E1.31 layer vector set to VECTOR_E131_EXTENDED_DISCOVERY:	
6.4.2 E1.31 Universe Discovery Packet: Source Name	- The source name must be null-terminated:	
	- The source name of a component must match the UACN field as specified in EPI 19 [ACN]:	
	- The source name may be the same across multiple universes sourced by the same component:	
	- The source name should be unique:	Left to the implementer / user-configuration
6.4.3 E1.31 Universe Discovery Packet: Reserved	- Octets 108-111 of the E1.31 Universe Discovery Packets are reserved for future use and must be transmitted as 0:	test_discovery_packet_transmit_format
	- Octets 108-111 of the E1.31 Universe Discovery Packets must be ignored by receievers:	test_discovery_packet_arbitary_reserved_parse
6.5 Processing by Receivers	- Receievers must discard packets if the receieved value is not VECTOR_E131_DATA_PACKET, VECTOR_E131_EXTENDED_SYNCHRONIZATION or VECTOR_E131_EXTENDED_DISCOVERY:	test_discovery_packet_unknown_framing_vector_parse, test_sync_packet_framing_layer_unknown_vector, test_malformed_data_packet_framing_layer_wrong_vector_parse
	- Receivers that do not support universe synchronisation may ignore packets with VECTOR_E131_EXTENDED_SYNCHRONISATION:	Doesn't apply as implementation supports universe synchronisation.
6.6.1 Transmission Rate	- E1.31 sources must not transmit packets for a given universe number at a rate which exceeds the maximum refresh rate specified in E1.11 [DMX] unless configured by the user to do so:	Left to the implementation using the library
	- E1.11 places special restrictions on the maximum rate for alternate START Code packets in Section 8.5.3.2:	
6.6.2 Null START Code Transmission Requirements in E1.31 Data Packets	- Transmission of Null START code data should only be done when it changes:	
	- Before entering this period of transmission suppression three packets of the values should be sent:	
	- During transmission suppression a single keep-alive packet should be transmitted at intervals of between 800mS and 1000mS, each keep-alive packet should have identical content to the last Null START Code data packet sent (but with sequence number still incremented normally):	
	- These requirements do not apply to alternate START code data:	
6.7.1 Network Data Loss	- Network data loss is a conditional defined as the absence of reception of E1.31 packets from a given source for a period of E131_NETWORK_DATA_LOSS_TIMEOUT:	test_source_1_universe_timeout
	- or the receipt of a packet containing the options field Stream_Terminated set to 1:	
	- Data loss is specific to a universe not a source, a specific universe is considered disconnected on data loss:	test_source_2_universe_1_timeout
6.7.1.1 Network Data Loss and Universe Discovery	- Sources experiencing a network data loss condition must reflect the change in the E1.31 Universe discovery list of universes no later than 2 E131_UNIVERSE_DISCOVER_INTERVAL's	Left to the implementation using the library to de-register a source that it is no longer sending on because it has lost its upstream source of data
6.7.2 Sequence Numbering	- Receivers that do not support sequence numbering of packets should ignore these fields:	Sequence numbering is supported, see below
	- Receivers that support sequence numbering should evaluate sequence numbers seperately for each E1.31 packet type and within each packet type seperately for each universe:	test_sequence_number_packet_type_independence, test_data_packet_sequence_number_univers
	- Receivers should process packets in the order received unless the sequence number of the packet receieved minus the sequence number of the last accepted sequence number is less than or equal to 0 but greater than -20:	test_data_packet_sequence_number_below_expected, test_sync_packet_sequence_number_below_expected, test_data_packet_sequence_number_exhaustive, test_sync_packet_sequence_number_exhaustive
7 DMP Layer Protocol	- DMP data should only appear in E1.31 Data Packets and not E1.31 Sync or Discovery packets	sync_parse_tests, discovery_parse_tests
	- The DMP data should be formatted as specified in Table 7-8	data_parse_tests
7.1 DMP Layer: Flags & Length	- The PDU length is encoded at the low 12 bits:	test_malformed_data_packet_dmp_layer_too_high_length_parse, test_malformed_data_packet_dmp_layer_too_low_length_parse
	- 0x7 must appear in the top 4 bits:	test_malformed_data_packet_dmp_layer_wrong_flags_parse
	- The DMP layer PDU length is computed starting at octet 115 and ends including the last value in the DMP PDU (octet 637 for a full payload):	test_malformed_data_packet_dmp_layer_too_high_length_parse, test_malformed_data_packet_dmp_layer_too_low_length_parse

7.2 DMP Layer: Vector	- The DMP layer vector must be set to VECTOR_DMP_SET_PROPERTY, receivers should discard packets if the receieved value is not VECTOR_DMP_SET_PROPERTY:	test_malformed_data_packet_dmp_layer_wrong_vector_parse
7.3 Address Type and Data Type	- The DMP layer address type and data type must be 0xa1, receivers must discard packets if the value is not 0xa1	test_malformed_data_packet_dmp_layer_wrong_address_data_parse
7.4 First Property Address	- The DMP Layers first property address must be 0x0000, receivers must discard packets if the value is not 0x0000:	test_malformed_data_packet_dmp_layer_wrong_first_property_address_parse
7.5 Address Increment	- The DMP layer address increment must be 0x0001, receivers must discard packets if the value is not 0x0001:	test_malformed_data_packet_dmp_layer_wrong_address_increment_parse
7.6 Property Value Count	- Must contain the number of DMX512-A [DMX] slots including the START code slot:	test_malformed_data_packet_dmp_layer_too_high_property_count_parse, test_malformed_data_packet_dmp_layer_too_low_property_count_parse
7.7 Property Values (DMX512-A Data)	- The first octet of the property values field is the DMX512-A START Code, The maximum number of data slots excluding the START Code is 512 data slots:	test_termination_packet_full_property_values_parse, test_malformed_data_packet_dmp_layer_too_high_property_count_parse
	- Alternate START Code data much be processed in compliance with ANSI E1.11 [DMX] Section 8.5.3.3: "DMX512 processing devices or any device that receives and re-transmits DMX512 shall state in the manual for the product how they process Alternate START Code packets. The acceptable processing methods are: 1) Block all packets containing particular Alternate START Codes. The START Codes blocked shall be declared (and may be all Alternate START Codes). 2) Pass unmodified all packets containing particular Alternate START Codes. The START Codes passed shall be declared. 3) Process the information contained in packets containing particular Alternate START Codes. The algorithm shall be declared in enough detail to allow the user to decide if the device will satisfy their needs. DMX512 in-line repeating transmitters shall not pass some packets with a particular Alternate START Code while blocking other packets containing the same Alternate START Code unless doing so as part of a stated processing algorithm."	Left to the implementation using the library, alternative start-code data is treated the same as any other start-code data within the implementation allowing the user of the library to choose how to handle the payload. test_send_recv_single_universe_alternative_startcode_multicast_ipv4 test_send_recv_single_universe_alternative_startcode_multicast_ipv6
8 Universe Discovery Layer	- The packet must be formatted as specified in Table 8-9:	discovery_parse_tests
8.1 Flags and Length	- The PDU length is encoded in the low 12 bits:	test_discovery_packet_discovery_layer_length_too_short_parse, test_discovery_packet_discovery_layer_length_too_long_parse
	- 0x7 must be encoded in the top 4 bits:	test_discovery_packet_discovery_layer_wrong_flags_parse
	- The PDU length is computed from octet 112 upto and including the last universe in the universe discovery PDU (octet 1143 for a full payload):	test_discovery_packet_discovery_layer_length_too_short_parse, test_discovery_packet_discovery_layer_length_too_long_parse
8.2 Universe Discovery Layer: Vector	- The university discovery layer vector must be VECTOR_UNIVERSE_DISCOVERY_UNIVERSE_LIST, receievers should discard packets if the received value is not VECTOR_UNIVERSE_DISCOVERY_UNIVERSE_LIST:	test_discovery_packet_discovery_layer_vector_unknown_parse
8.3 Page	- Indicates the page being specified in the set of universe discovery packets starting at 0:	test_discovery_packet_page_higher_than_last_page_parse
8.4 Last Page	- Indicates the index of the last page in the set of universe discovery packets:	test_discovery_packet_page_higher_than_last_page_parse
8.5 List of Universes	- Must be numerically sorted:	Taken to mean numerically sorted in accending order with lower universe numbers at a lower position (octet) within the packet. test_discovery_packet_random_order_parse, test_discovery_packet_decending_order_parse
	- May be empty:	test_discovery_packet_no_universes
	- Should contain all of the universes upon which a source is actively transmitting	test_universe_discovery_multiple_universe_one_source_ipv4
E1.31 Data and Synchronisation information:		
9 Operation of E1.31 in IPv4 and IPv6 Networks	- The standard can work over either and which modes are supported should be indicated:	Implementation supports either IPv4 or IPv6 test_send_recv_single_universe_multicast_ipv4, test_send_recv_single_universe_multicast_ipv6
9.1 Association of Multicast Addresses and Universe	- The standard should work over multicasting	test_send_recv_single_universe_multicast_ipv4, test_send_recv_single_universe_multicast_ipv6
	- The standard should also work using unicast	test_send_recv_single_universe_unicast_ipv4, test_send_recv_single_universe_unicast_ipv6
	- Addressing of multicast traffic done by setting 2 least significant bytes to the desired universe number or synchronisation address:	test_universe_to_ip_ipv4_both_bytes_normal, test_universe_to_ip_ipv6_both_bytes_normal
	- Sources operating over IPv4 and IPv6 simultaneously should transmit identical E1.31 packets regardless of IP transport used:	test_ip_equivalence
	- Recievers operating in IPv4 and IPV6 simultaneously should not process E1.31 packets differently based on the IP transport:	The library passes data up without specifying IP version used with same recv() usage regardless of ipv4 or ipv6 which shows no difference dependent on IP version. test_discover_recv_sync_runthrough_ipv6, test_ansi_e131_appendix_b_runthrough_ipv6, test_discover_recv_sync_runthrough_ipv4, test_ansi_e131_appendix_b_runthrough_ipv4
	- Receivers operating in IPv4 and IPv6 simultaneously seeing the same packet via both IP transports shall only act on one instance of that packet:	Receiver only operates in one IP version at once, the user of the library can use 2 receivers in different ip version simultaneously but it is left to them to only act on one instance of the packet.
9.1.1 Multicast Addressing	- E1.31 devices should not transmit on address 239.255.255.0 through 239.255.255.255:	test_universe_to_ip_ipv4_limit_high
	- E1.31 devices shall not used universe number 0 or univere numbers [64000 - 65535] excluding universe 64214 (used for universe discovery only):	
	- The identity of the universe must be determined by the universe number in packet and not assumed from multicast address:	test_send_recv_wrong_multicast_universe
	- E1.31 devices should also respond to E1.31 data receieved on its unicast address:	test_send_recv_single_universe_unicast_ipv4
	- When multicast addressing is used the UDP destination port shall be set to the standard ACN-SDT multicast port ACN_SDT_MULTICAST_PORT:	test_universe_to_ip_ipv4_both_bytes_normal, test_universe_to_ipv4_lowest_byte_normal, test_universe_to_ip_ipv4_limit_high, test_universe_to_ip_ipv4_limit_low
	- For unicast communication the ACN-SDT multicast port shall be used by default but may be configured differently	test_send_recv_single_universe_unicast_ipv4

9.2 Multicast Subscription	- Receivers supporting IPv4 must support IGMP v2 or any subsequent superset of IGMPv2's functionality:	Provided by underlying Socket2 rust library
	- Receivers supporting IPv6 shall support MLD V1 or any subsequent subset of MLD1's functionality:	Provided by underlying Socket2 rust library
9.3.1 Allocation of IPv4 Multicast Addresses	- Multicast IPv4 addresses must be defined as in Table 9-10	test_universe_to_ipv4_lowest_byte_normal, test_universe_to_ip_ipv4_both_bytes_normal, test_universe_to_ip_ipv4_limit_high, test_universe_to_ip_ipv4_limit_low
9.3.2 Allocation of IPv6 Multicast Addresses	- Multicast IPv6 addresses must be defined as in Table 9-11 and Table 9-12	test_universe_to_ipv6_lowest_byte_normal, test_universe_to_ip_ipv6_both_bytes_normal, test_universe_to_ip_ipv6_limit_high, test_universe_to_ip_ipv6_limit_low
9.4 IPv4 and IPv6 Support Requirements	- E1.31 sources need to be able to operate on both IPv4 and IPv6 potentially simultaneously:	
	- The state of IPv4 / IPv6 operation should be configurable by the end user:	Library allows providing IPv4 or IPv6 address to decide. test_send_recv_single_universe_multicast_ipv4, test_send_recv_single_universe_multicast_ipv6
10.1.1 Boot Condition	- A DMX512-A [DMX] to E1.31 translator shall not transmit E1.31 data packets for a given universe until it has received at least one valid DMX512-A input packet for that universe:	Left to the user of the library, this library may or may not be used as part of a DMX512-A translator
10.1.2 Temporal Sequence	- A DMX512-A [DMX] to E1.31 translator shall transmit packets in the order in which they were received from the DMX512-A source:	
10.1.3 Loss of Data	- On loss of data as defined in DMX512-A a source shall terminate transmission as per Section 6.7.1:	
10.2.2 Loss of Data	- There must be an operating mode where upon detection of loss of data as defined in 6.7.1 for all sources of a universe a source shall immediately stop transmitting DMX512-A packets:	This is out of the scope of the library and is upto the user of the library (if they are transmitting DMX packets)
11 Universe Synchronization	- There is no restriction on the number of synchronisation addresses allowed on a single network:	Implementation enforces no restriction except for the limit of possible universes allowed by the protocol [0, 63999]
	- It is possible to have multiple independent universes configured for E1.31 synchronisation concurrently:	test_send_recv_multiple_sync_universes
11.1.1 When to Begin Synchronizing Data	- A receiever should begin universe synchronisation upon receipt of the first syncronisation packet for that universe:	test_send_recv_across_universe_multicast_ipv4, test_send_recv_across_universe_multicast_ipv6
11.1.2 When to Stop Synchronizing Data	- A receiever should stop universe synchronisation if it does not receieve an E1.31 synchronisation packet on that universe within E131_NETWORK_DATA_LOSS_TIMEOUT:	test_send_sync_timeout
	- The behaviour on timeout may be determined by the Force Synchronisation Option bit:	Not implemented in this library
11.2.1 Arrival of Multiple Packets Before Processing	- An E1.31 receiever should only synchronise using the definitive E1.31 data for that universe:	
	- If there is a single source the definitive data is the data packet with the most recent valid sequence number:	
	- If there are multiple active synchronisation sources on the same synchronisation address it is beyond the scope of the standard:	
11.2.2 Delays Before Universe Synchronization	- Recommended to add a configurable delay between data packets and transmission of an E1.31 synchronisation packet:	Left to the user of the library
12 Universe Discovery	- Legacy devices may not implement it even though to be compliant they should:	The library does not rely on other sources implementing universe discovery, data send/recv works without requiring it test_send_recv_single_universe_multicast_ipv4, test_send_recv_single_universe_multicast_ipv6
12.1 Universe Discovery and Termination	- A source that is not sending any universe data may stop sending E1.31 Universe Discovery Packets until transmission resumes or alternatively a source could send an empty list of universes:	The implemention keeps sending discovery packets with an empty list of universes, test_universe_discovery_no_universes_ipv4
12.2 Termination of Stream Transmission	- A E1.31 data stream is terminated when either a Stream_Terminated packet is receieved:	
	- or if no packet is receieved for an interval of E131_NETWORK_DATA_LOSS_TIMEOUT:	
	- A source that has terminated transmission for an E1.31 universe must reflect the change no later than the end of the second E131_UNIVERSE_DISCOVERY_INTERVAL	Left to the user of the library to indicate that it has terminated transmission by deregistering the universe from the sACN source provided by the library
Appendix A: Defined Parameters (Normative)	- All parameters used must match those specified in Appendix A:	check_ansi_e131_2018_parameter_values
B.1 Universe Synchronization for Sources	- The completed implementation must produce exactly the example response given for the given conditions / inputs:	Example walked through in the below test, including noting differences between the example and actual due to force_synchronisation not being implemented test_ansi_e131_appendix_b_runthrough
B.2 Universe Synchronization for Receivers	- The completed implementation must produce exactly the example response given for the given conditions / inputs:	