ClimateTalk 2.0
Generic Application Specification

Document revision: 01
Release: June 12, 2013

Abstract

ClimateTalk is a universal language for innovative, cost-effective solutions that optimize performance, efficiency and home comfort. The ClimateTalk Open Standards define a set of messages and commands to enable interoperability, enhanced user interface, and machine to machine control independent of the physical layer connecting the devices.

This document defines the application requirements corresponding to OSI Layer 7 that are generic for any subsystem operation and interaction with other devices on a ClimateTalk network.

Updates

This specification may be updated at any time and may be superseded by a more recent version or amended to from time to time. Users should be certain they are using the current ClimateTalk version and the latest revision of the documents.

The released versions of all specifications are available at http://www.ClimateTalk.org

Version History

ClimateTalk Version	Document Revision	Release Date	Comments
V 0.9		2008-11-07	Pre-Release
V 1.0		2009-08-24	Initial Release
V 1.1		2011-06-23	Errata Package
V 1.3		2011-11-02	Additional Errata Updates, Revised Formatting
V 2.0	00	2013-01-18	Version 2.0 Release – Update send methods, node list, shared data to support Version 2.0 devices. Add Zone Controller to routing priority table.
V 2.0	01	2013-06-12	Updated descriptions for Send Methods to refer to applicable Networking Specification. Updated 6.4.2 Network Node List Example and Framing to explicitly state the entire node list must be checked for active nodes.

Contributors

The following is a list of ClimateTalk Alliance member companies that were actively engaged in the development of this standard:

A.O. Smith Water Products Company

ecobee inc.

EDC

Emerson Electric, Co.

EWC Controls, Inc.

Microchip Technologies, Inc.

Nogginhaus, LLC.

Research Products Corp.

Rheem Manufacturing Company

Zonefirst

Table of Contents

List of Figures

List of Tables

1.0 Overview

1.1 ClimateTalk Model

ClimateTalk is an open standard that defines a set of messages and commands to enable interoperability, enhanced user interface, and machine to machine control independent of the physical layer connecting the devices.

The messages and commands defined by ClimateTalk Information Model (CIM) are the presentation and application layers as defined by the OSI Model[1]. ClimateTalk Applications are fully defined at Layer 7 of the OSI model by a combination of a Device Specific Application Profile, the Generic Application Specification and the Command Reference.

ClimateTalk messages can be carried over any physical medium following the OSI model. The ClimateTalk Presentation Layer defines how messages are executed over the various physical mediums in use.

CT-485 and CT-LWP are wired serial physical and network layers designed to support the formation of ClimateTalk networks and transport ClimateTalk messages, but other OSI based protocols – including wireless transports - can be used as well.

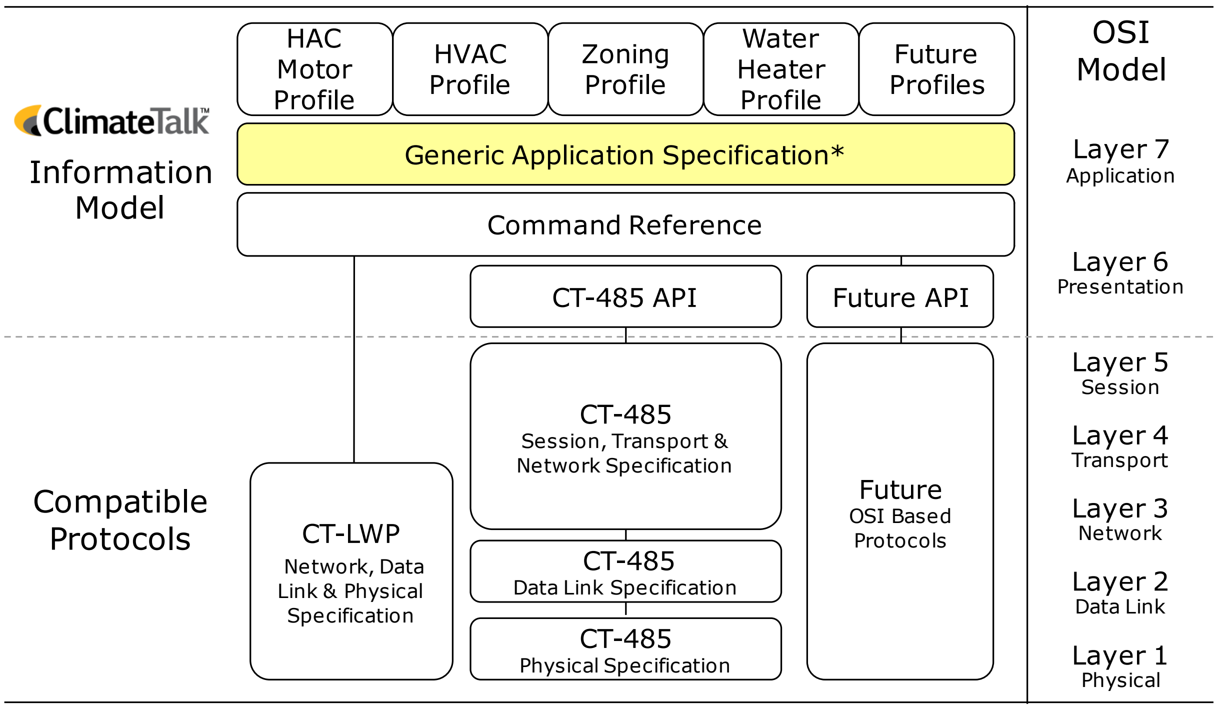
1.2 Scope

This document defines the generic application requirements for all devices designed to ClimateTalk Open Standards. This profile defines how all devices interact with other devices on a ClimateTalk network. Profiles begin with the requirements in this document and expand on them to satisfy the needs of their application.

The ClimateTalk Open Standards package shown in Figure 1 - OSI Layers for ClimateTalk Implementation prescribes the mandatory requirements to ensure proper network formation of interoperable devices. Membership in the ClimateTalk Alliance as well as successful completion of mandatory conformance testing is required for listing a product as a ClimateTalk Certified Device.

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269

Figure 1 - OSI Layers for ClimateTalk Implementation



* This document

This profile also defines the testable requirements used to validate that a device is behaving properly within a ClimateTalk network. Each device must comply with the mandatory requirements defined in this document as well as all other ClimateTalk standards applicable to the device functionality.

2.0 Normative References

A good understanding of the most recent version of the following documents is required to apply the contents of this specification correctly.

ClimateTalk Generic Device Specific Application Profiles

- *HVAC Application Profile*
- *HAC Motor Application Profile*
- *Zoning Application Profile*
- *Water Heater Application Profile*

ClimateTalk Command Reference

ClimateTalk CT-485 Application Protocol Interface

ClimateTalk CT-485 Networking Specification

ClimateTalk CT-485 Data Link Specification

ClimateTalk CT-485 Physical Specification

ClimateTalk CT-LWP Specification

3.0 Terminology

3.1 Definitions

Cluster	Level 2 and 3 menus combined; a Sub Menu item and all possible options for its value.
Cold Start	A start of a subsystem from a state of not being powered to being powered.
Control ID	Unique 2-byte Identifier assigned to each device by the ClimateTalk Alliance that is used to identify the control application.
Data Tags	User Menu tags intended to mark a numerical value parameter.
Embedded Tags (eTags)	Supplemental and Reserved User Menu Tags that allow functionality beyond basic HTML Screen formatting.
Fetch	Request sent by client to retrieve User Menu information from a subsystem server. This server responds with the requested User Menu information.
Manufacturer ID	Unique 2-byte Identifier assigned to each manufacturer by the ClimateTalk Alliance that is used to identify the subsystem.
Network Coordinator	ClimateTalk device which establishes the network and through which messages are routed.
Null Byte	Hexadecimal zero, also shown as 0x00.
Off-Board Bus Interface (OBBI)	Communicating HVAC systems rely that devices all communicating on the same physical network bus, i.e. CT485. For the cases in which non-communicating HVAC equipment exists, the bus requires an interface module to be the translator between old HVAC and new communicating HVAC devices. An OBBI will take serial traffic and convert it to normal 24VAC thermostat output.
Post	Client sends updated User Menu information to a subsystem. This server responds to the client with the updated User Menu information.
Profile	Set of rules governing the implementation of certain aspects of the protocol, including timing rules and communication rules in order to function properly.

Refresh	Information encoded in the User Menu that tells the application how often to update the User Menu by performing another Fetch. Refresh tags are found in a Level 1 Request.
Stealth	A filtering technique where servers are aware of specific content while clients are restricted.
Warm Start	A re-start of a device from a state where the device needs to return to idle and relearn the network. This is defined as receiving a network Node List.
Web Page	The Micro Tagged HTML User Menu bytes that reside on the subsystem. The Web Page contains all User Menus, Variable Pointers and Attributes for viewable and selectable structures in the CTUM environment.

3.2 Acronyms

CT	ClimateTalk
CTUM	ClimateTalk User Menu
HTML	Hyper Text Markup Language
IEEE	Institute of Electrical and Electronics Engineers
TBC	To Be Calculated

3.3 Word Usage

The conventions used in this document are modeled after the definitions of the 2009 IEEE Standards Style Manual. The IEEE Standards Style Manual may be obtained from http://standards.ieee.org/guides/style/.

can	Equivalent to *is able to* or *is capable of*.
may	Equivalent to *is permitted to* or *is allowed to*. The use of *may* means that something is optional, and does not imply a requirement.
must	Used to describe situations where no other course of action is possible.
shall	Equivalent to *is required to*. Use of the word *shall* means that the specification shall be implemented exactly as described in order to ensure correct operation and interoperability with other devices.
should	Equivalent to *is recommended that*. This is used in situations where there are several possible options, but one option is preferable to the others.

4.0 Warm and Cold Start Procedures

4.1 Types of Start Up

Two types of start-up are defined:
1. Cold Start
2. Warm Start

A cold start occurs when the device transitions from a state of not being powered to being powered.
A warm start is considered any condition where a device needs to return to idle and relearn the network.

4.2 Cold Start Procedure

All devices shall do the following:

1. Start up in default state.
2. Wait to receive a network Node List. On receiving the Node List, do Warm Start Procedure.

4.3 Warm Start Procedure

Warm Start Procedure – Unless noted otherwise in device profile.

1. Check the network Node List. Am I on the list?
    a. If no, stop and wait.
    b. If yes, continue on with the warm start procedure.
2. Device is now considered to be on the network.

5.0 Frame Format

The ClimateTalk Message Frame can be classified into 4 basic segments: Message Header, Packet Header, Packet Payload and Message Footer. The physical, datalink and network layers are responsible for the message header and footer, and the application layer is responsible for the contents of the Packet Header and Packet Payload.

Table 1 – ClimateTalk Message Structure Elements

Element	Segment	Size (in bytes)
Addressing and Routing	Message Header	Variable
Message Type	Packet Header	1
Packet Number		1
Payload Length		1
Packet Payload	Packet Payload	0-240
Message Checksum	Message Footer	Variable

The size of the message header and footer depend on the type of network being used to carry ClimateTalk messages.

5.1 Message Header Elements

Message Header Elements provide information about the packet embedded within a ClimateTalk message. Information on what type of packet, who sent the packet, who should receive the packet and how the packet should be sent are all contained within this segment of the message.

5.2 Packet Header Elements

Packet Header Elements provide additional information about the packet embedded within a ClimateTalk Message Frame.

ClimateTalk only allows one packet to be contained within a ClimateTalk Message Frame and does not allow padding the message frame with any additional data.

By processing the Packet Header as part of the Message Frame integrity check, the security of the system is greatly increased by providing a secondary integrity check that validates the message frame size and reduces the vulnerability to piggy backing data within a Message Frame.

5.2.1 Message Type

The Message Type within the packet header is filled by the original sending node with the Message Type for the packet being sent. The Message Type tells the receiving node what type of data is being sent.

There is only one byte allocated within a ClimateTalk Message to represent the Message Type. This means that only 256 possible unique Message Types can be represented.

To provide for the requirement that all request message types maintain a corresponding response message type, the most significant bit of the Message Type is reserved for flagging response messages. This effectively reduces the maximum range of unique messages to 128.

Those 128 unique Message Types have been further divided into ranges to support the evolution and growth of ClimateTalk. Consult the Reference Specifications for allocation of Message Type Ranges as well as specific Message Types in use.

Checking the most significant bit of the Message Type determines if particular message is a Request Message or a Response Message. If the D7 Bit of the Message Type is CLEAR then the message is a Request Message and if the D7 Bit is SET then it is a Response Message.

Table 2 – Message Type Bit Decoding

Message Type – Bit Decoded							
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Response Bit	Unique Message Value						

5.2.2 Packet Number

The Packet Number field is defined as shown in Table 3 - Packet Number Field.

Table 3 - Packet Number Field

Packet Number – Bit Decoded							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Dataflow Flag	**Reserved**	**Version**	**Reserved**				
Set for R2R and ACK	**0**	**1/0**	**0**	**0**	**0**	**0**	**0**

Bit 7 is used as the “Dataflow Flag” and shall be set to 1 for any R2R or Acknowledgement messages.

Bit 6 and Bits 4-0 are reserved for future use.

Bit 5: A CT2.0 coordinator shall set the Version bit to a '1' when sending Node Discovery Request messages. A CT2.0 subordinate shall reply with its version bit clear, '0', indicating it is a CT2.0 device. A CT1.0 device shall reply with its version bit set indicating it is a CT1.0 device. The setting of the version bit by the coordinator is required to obtain the expected node discovery response from some older CT1.0 devices which echoed the version bit during the addressing sequence. A coordinator shall ignore this bit except for CT version determination during a node discovery response.

5.2.3 Payload Length

The Payload Length is the length of the payload that is contained in the Payload Data portion of the packet. Since the amount of payload can be of variable length it is necessary to know the intended length of the data.

A message frame that is received can be validated to contain a packet of an expected length based on the payload length and the message frame's total size. The Payload Length plus the Message Header Length should always resolve to the expected end of the message frame. Payload Lengths that are too long indicate that the entire message was not received or was inappropriately formatted, therefore are not processed. Payload Lengths that are too short are not processed since they break the rule that a Message Frame should contain only one packet with no additional data.

5.3 Packet Payload Element

The Packet Payload Element provides the application a space to read data (on reception) or write data (when preparing a transmission). This information is the actual data being sent in a ClimateTalk Network by its users. This size of the Packet Payload Element is given by the Payload Length byte within the Packet Header Element.

5.3.1 Packet Payload Data

The Payload Data section of the ClimateTalk packet structure is the application data that has been encapsulated within a ClimateTalk packet. This data must match the length provided by the Payload Length byte within the packet header.

5.4 Message Footer Element

The Message Footer provides a checksum of the entire message frame. The exact contents will be determined by the type of physical medium being used to transmit the ClimateTalk message. The Message Footer shall reside at the expected end of the Message Frame based on the payload length in order to align the checksum at the expected location for correct processing. This further protects from processing corrupted data streams.

6.0 Application Interface

6.1 Node types

Each device is assigned a Node Type. The Node Type may be used to determine what devices are present on the network and therefore what functions the system is capable of performing.

Refer to the Appendices in the Command Reference for a list of all currently defined Node Types. If a Node Type is not identified in that table, a request to the CIM WG can be made to add it.

6.2 Send Methods

For the application to inform the network who the intended recipient is, a mechanism known as send methods is used in ClimateTalk. A send method has two send parameters which are used in conjunction with the send method by the networking infrastructure to determine the intended destination for the message. The use of send methods is detailed in the following subsections.

On any message passed by the application on to networking, it includes the send method and send parameters to guide the networking layer on how and where to deliver the message. Conversely, on all packets received and passed up to the application, the networking layer will pass up the send methods and send parameters received with the message.

For any of the send methods, if no device is found matching the specified send parameters for routing, then the message will not be routed, and will simply be discarded by the networking layer.

6.2.1 Send Method 0: Non-Routed Messages

Refer to applicable *ClimateTalk Networking Specification* for information regarding this send method.

6.2.2 Send Method 1: Routing by Priority Control Command Device

Refer to applicable *ClimateTalk Networking Specification* for information regarding this send method.

6.2.3 Send Method 2: Routing by Priority Node Type

Refer to applicable *ClimateTalk Networking Specification* for information regarding this send method.

6.2.4 Send Method 3: Routing by Socket

Refer to applicable *ClimateTalk Networking Specification* for information regarding this send method.

6.3 Shared Data

Shared Data is the term used for a device's configuration data. Shared Data can be redundantly stored by other ClimateTalk devices on the network and retrieved from the network in the absence of valid Shared Data within a device. Backing up configuration data in this way makes it easier to replace devices in the field, as the devices can retrieve their configuration parameters from the network.

Only certain devices are allowed to request network storage of Shared Data. Those devices are as follows: Thermostat, Zone Controller, Air Handler, Gas Furnace, Air Conditioner, Heat Pump, and Unitary Control. The following sections describing Shared Data are only applicable for those devices listed if that device needs to be factory programmed with Shared Data or wishes to request network storage of Shared Data, or is capable of becoming a coordinator.

6.3.1 Shared Data Record Description

The Shared Data record for a device shall be a maximum of 200 bytes in length. A Shared Data record consists of two parts: a header defined by this specification and the remainder defined by the manufacturer.

6.3.1.1 Application Shared Data Header

The first 6 bytes of a maximum 200 byte long Shared Data record are specified below. These fixed fields consist of the total Shared Data record length, Node Type of the device to which the data belongs, and the manufacturer ID of the device/subsystem. Control ID and Manufacturer ID shall be represented as follows: If company A provides a circuit board to company B for the use in a company B subsystem, the Control ID would be the ClimateTalk OEM ID for company A and the Manufacturer ID would be the ClimateTalk OEM ID for company B.

Table 4 - Application Shared Data Detail

Shared Data Length	Control ID	Manufacturer ID	Application Node Type	App Data
1 Byte	2 Bytes	2 Bytes	1 Byte	n - 6 Bytes
n	Assigned by ClimateTalk Alliance	Assigned by ClimateTalk Alliance	From Node Type Table	

6.3.1.2 Manufacturer-Defined Shared Data

The manufacturer-defined portion of Shared Data is a maximum of 194 bytes in length. Manufacturers may use this space as they see fit. It is, however, recommended that the manufacturer portion of Shared Data include a) factory-defined fields that cannot be changed by the user and b) a user-modifiable field that could be changed via options like a user menu.

The factory-defined section may contain default user menu values, allowing a field service technician to restore factory default settings to the user-modifiable section. Refer to Annex B – Sample Shared Data Structure for a possible arrangement when using factory-defined and user-modifiable sections.

6.3.2 Initialization

Shared Data is initialized at the factory using a Set Factory Shared Data to Application message. Table 4 illustrates the data payload that would be sent using this message type.

A Get Shared Data from Application message may be used to verify that the right data has been stored in the device. Consult the applicable API Reference for more information on the commands used to initialize Shared Data.

6.3.3 Network Shared Data Storage

All ClimateTalk devices that are capable of becoming a coordinator must provide the service of storing Shared Data for devices that make the storage request. A coordinator capable device shall also distribute any Shared Data it receives to other devices on the network.

Any ClimateTalk device may optionally provide a service to store Shared Data for devices that are capable of requesting network storage of Shared Data. The coordinator's Shared Data storage and replication responsibilities are covered in more detail in the Network Specification.

A device may make a request for the network storage of its Shared Data by issuing message Set Application Shared Data to Network. The following table illustrates the data payload that would be sent using this message.

Table 5 - Network Shared Data Storage Detail

Sector Node Type	**Application Shared Data Detail**				
1 Byte	1 Byte	2 Bytes	2 Bytes	1 Byte	n - 6 Bytes
From Node Type Table	Reference Table 4				

Message, Get Application Shared Data from Network, may be used to retrieve network Shared Data. Consult the applicable API Reference for more information on the format of commands used for shared data storage.

6.3.4 Shared Data Sources

Four possible non-volatile/permanent storage areas for storage of Shared Data are noted below:

Memory card/stick Shared Data	= MC (optional)
Application microcontroller or EEPROM Shared Data[2]	= ADEE
Motor Scratchpad	= MTR (optional)
Network Shared Data	= ND

An application shall inspect the available Shared Data storage areas in sequential order with the Memory Card being checked first and the Network Shared Data checked last. If any data retrieved is not valid or the storage source is not supported, the application shall check the next storage location.

6.3.5 Shared Data Validity

Shared Data is checked for validity to determine if the following is correct:

- Correct Control ID
- Correct Manufacturer ID
- Correct Unit/Node ID
- Number of Shared Data bytes does not exceed limit (200)
- Valid checksum for Shared Data
- Shared Data conflicts with installed components: Shared Data must match any available external data that can be used to determine if Shared Data is valid for the system. For example, indoor HVAC controls can request a communicating indoor blower motor to report its horsepower or outdoor controls can query a communicating compressor for its tonnage which can then be verified against Shared Data. It is conceivable to provide the application a method to retrieve the model number of the unit it controls and to verify this against the Shared Data.

Note that a shared data set as a whole is rejected or accepted. An application shall not accept and use part of the Shared Data while deeming another part invalid. Two Shared Data records may not be combined to obtain valid Shared Data.

6.3.6 Shared Data Rules to Ensure Integrity

1. Any validated set of Shared Data written to the device with a Set Factory Shared Data to Application message shall be written to all available storage locations regardless of prior content.

2. Every power-up and reset, the memory card Shared Data shall be verified, and if deemed valid the contents of the ADEE, MTR, and ND shall be checked and/or updated to match the contents of the memory card.

[2] The ADEE non-volatile storage is physically part of the device and not able to be removed.

The above rule provides an installer the option for a field upgrade of Shared Data by using a memory card.

However, since the ADEE is defined as being part of the device and a memory card may accidentally become disconnected, an application can attempt to recover Shared Data changes that may have occurred with the device's own memory card disconnected. Unless explicitly called out in an application requirement, the following is the only method by which the application can initiate a write of an entire shared data set to the memory card without first receiving a Set Factory Shared Data to Application message.

> The optional recovery method explained below applies only for controls that support a memory card and have organized shared data with factory-defined and user-modified sections.
>
> When factory data is written to the device in the presence of an installed memory card, the application writes the factory data to the memory card after validation, regardless of its prior content. A pseudo-random number is also written to the MC and ADEE to associate a specific Memory Card with a specific device. This number shall be referred to as the MC Identifier.
>
> If the device is in possession of valid Shared Data in both its ADEE and MC at startup, and the factory-defined area of the MC data is found to be identical to the factory-defined area of the ADEE, and the user-modifiable Shared Data is different, the Application checks the MC Identifier. If the MC Identifier matches, the Application replaces the MC shared data with ADEE shared data. In addition, MTR and ND are verified to hold the same data, and replaced if it is different. Note: The MC Identifier is only checked if the factory-defined Shared Data is the same (MC vs. ADEE) and the user-modifiable Shared Data is different (MC vs. ADEE).

3. If the memory card Shared Data is not valid or the memory card is not supported, the application shall inspect the next Shared Data source in the order of listed priority, ADEE, MTR, and ND. When valid Shared Data is located, the three sources ADEE, MTR, and ND shall be checked and/or updated with the valid Shared Data.

4. Whenever shared data changes, the application shall update MC, ADEE, MTR, and ND with this new shared data.

5. If the device is equipped with a resident memory card (one intended to stay connected to the device), it shall be physically attached to the unit. This means manual intervention is required to remove it or both a memory card and control board for use on another system. Proper training and warning labels will be required to prevent removing a memory card from the sheet metal.

6. For devices that support a memory card: To discourage an installer from inserting or removing the memory card with power applied to the device, the

memory card contents shall only be checked for valid shared data as part of a power-up or reset condition.

7. For devices that support a memory card: If the memory card contents cannot be read or are determined invalid during power-up or reset, the application will not attempt to update the memory card, but will instead continue to operate using available ADEE, MTR, and ND. Each power-up or reset the application can determine if the memory card data is valid for use.

6.3.7 Application Rules for Shared Data Activity on the Network

There are two messages that the Application uses to push validated Shared Data out to the network and retrieve valid Shared Data from the network. These are the Set Application Shared Data to Network and Get Application Shared Data from Network messages. For details on these message types and their packet organization, consult the API Reference.

On cold start, the application must determine whether or not it has valid Shared Data. If the application does have valid Shared Data it will issue a Set Application Shared Data to Network message to the network coordinator when it receives a node list. Additionally, the application will issue the Set Application Shared Data to Network message to the network coordinator after receiving a new node list.

If the application does not have Shared Data or if it has invalid Shared Data, it will issue a Get Application Shared Data from Network message to acquire Shared Data from the network coordinator. If this attempt fails (the coordinator does not have the requested record) then the application will retry after every timeout period (defined per profile) or when it receives a new node list.

On any internally or externally caused change to an application's Shared Data, the Set Application Shared Data to Network message shall be issued to update the network shared data.

6.3.8 Query of Application Shared Data

A device shall respond with the Shared Data set it is currently using in response to a Get Shared Data from Application message. Consult the applicable API Reference for more information on the format of this message.

If no Shared Data is present, a device shall respond with either a Payload Length of one (1) followed by the Node Type, or a Payload Length of zero. Either response shall be an indication the application does not have usable Shared Data. The preferred response when no Shared Data is present is a payload length of zero.

6.4 Network Node List

All devices are responsible for processing and responding to a Network Node List from the coordinator. All devices must accept a node list of up to 240 bytes.

6.4.1 Network Node List Update Notifications

The network coordinator has the responsibility for monitoring all devices on the network for changes to the network configuration. Any time the coordinator determines that a configuration change has taken place, it will stop normal network operations to send a new network node list to all devices on the network. The new network node list will contain the most up to date network configuration.

Possible reasons for a network configuration change include:

1. A new device was added to the network configuration.
2. A new session for a device.
3. A device was dropped from the network configuration.

Upon receiving a network Node List from the network, a device shall be capable of determining the actual network environment from the data payload. The list identifies the device's own network status and what other nodes are active on the network.

Upon seeing a Node Type corresponding to itself in the list, the device shall consider itself online.

If the device is online and receives a network Node List, the device will initiate a warm restart. See Section 4.0 for details.

6.4.2 Network Node List Example and Framing

The Network Node List contains a sequence of node types that when the value is not zero indicates the presence of a node of the type indicated by the non-zero value at a node list index. Presence of the same node type value at multiple positions will indicate the presence of multiple nodes of that type. The node list can be as big as the maximum payload size, and a subordinate application must be capable of receiving and processing it. To support future versions of the ClimateTalk standard, any node detected in a node list should be considered an active node on the network. Additional information regarding node lists is located in the *ClimateTalk Networking Specification*.

Below is an example of the payload data for a Network Node List for a network containing:

1. Three nodes of type '3'
2. One node of type '1'
3. One node of type '5'

Table 6 – Example Node List of length 64

Index	0	1	2	3	4	5	6	7	…	60	61	62	63

Node Type	3	1	5	0	3	0	3	0	…	0	0	0	0

6.5 Subsystem Installation Test

If a subsystem supports installation tests, it may support the ability to have the test be initiated by a remote device. The remote device sends a Subsystem Installation Test control command to the subsystem, requesting that it initiate the test. On receiving this command, the subsystem runs the test. The status and results of the test may be sent to the remote device or other display interface via the Set Display Message command.

The format of the Subsystem Installation Test control command and the Set Display Message command is in the ClimateTalk Command Reference document.

Subsystems are not required to support an installation test. If a subsystem does not support installation tests and it receives a request to perform one, it shall respond per paragraph 8.3 Unknown Application Payload.

The test may be ended by the subsystem itself or at any time by the remote device via the Subsystem Installation Test control command.

The remote device may be a thermostat or diagnostic tool or another subsystem. The ability of a particular device to display information to a user will vary depending on the type of display the device has. Devices with dot-matrix or other visually rich displays are required to support the initiation and monitoring of a Subsystem Installation Test. Devices with segmented or other visually poor displays are not required to support initiation and monitoring of a Subsystem installation test.

7.0 Message Data Interfaces (MDIs)

7.1 Manufacturer ClimateTalk ID

The Manufacturer ClimateTalk ID is the number provided to an adopter of the protocol to be used to generate manufacturer specific messages or closed communications.

The Manufacturer ID number is assigned by the standards committee and is known only by that group. Only the company to which the number is assigned has the right to use that number. Any use of a number without permission is in violation of the license agreement.

The Manufacturer ID number zero is used for open messages defined and utilized within the protocol. One example would be the shared data message type.

7.2 Database Identification (DB ID) Tag

It is not desirable to return the entire contents of an MDI every time a single piece of data is required. Database IDs are used to tag individual elements of the MDI so that they can be retrieved one at a time.

The ability for controls to block their data types allows for the ability of controls to either use that tag of information or to exclude it from their device. The ID number is used by the device making the request to understand if that device has that ability of what is required by that piece of information.

7.2.1 DB ID Datagram

This section details the bit-by-bit breakdown of how the DB ID tag is defined and set in relation to the MDI.

Figure 2 - DB ID Tag Bit-By-Bit Breakout

Type	Length	**DB ID TAG**	**DB LENGTH**	Value	CRC	
TBC	3	Number	Number	Data	TBC	TBC

$$\underbrace{\qquad\qquad}_{\text{DB ID}} + \underbrace{\qquad\qquad}_{\text{DATA}}$$

In this datagram, for every response, the data payload will have a corresponding DB ID Tag and DB Length followed by the data defined for the corresponding device DB ID.

7.2.2 DB ID Tag Example

Table 7 - DB ID Tag Example shows a sample sensor MDI that would be sent upon request of this devices sensor data. If a device only has two DB IDs defined due to it not being a full featured device, the response back from the request would only contain those defined DB IDs in the data payload.

Table 7 - DB ID Tag Example – Sample System Sensor MDI

DB ID Tag	DB ID Length	Byte	Bit	Description	Size	Element	Notes
0	2	0-1	15	Some Temperature Sensor	1 bit	Validity Flag	1 = Valid 0 = Invalid or Not Installed.
			14		1 bit	Signed Bit	1 = Negative 0 = Positive
			13-4		10 bits	Whole Value	Absolute value of the integer part of the temperature in °F 0 to 1023
			3-0		4 bits	Fractional Part	Absolute value of the fractional part of the temperature in °F 0 to 15 in 1/16th increments.
1							
2							
3							
4	2	0-1	15	Other Temperature Sensor	1 bit	Validity Flag	1 = Valid 0 = Invalid or Not Installed.
			14		1 bit	Signed Bit	1 = Negative 0 = Positive
			13-4		10 bits	Whole Value	Absolute value of the integer part of the temperature in °F 0 to 1023
			3-0		4 bits	Fractional Part	Absolute value of the fractional part of the temperature in °F 0 to 15 in 1/16th increments.

Figure 3 - DB ID Datagram Example demonstrates a response to a sensor request for the sample device where only two sensors are incorporated.

Figure 3 - DB ID Datagram Example

Message Type - Sensor response	Payload Length	DB ID Tag	DB Length			DB ID Tag	DB Length			CRC
0x87	0x08	0x00	0x02	Some Temp		0x04	0x02	Other Temp		TBC

7.3 Identification Message Data Interface (MDI)

All devices shall store the ClimateTalk specification version number, the software version number, software revision number, and unique serial number at the very minimum. Additional parameters are detailed in the Command Reference Document and the applicable application profile.

The optional part of the Identification MDI includes details such as installation date, address of installation, etc. that can be used during end of life or fault trend analysis.

All devices shall respond to an identification message request with a payload containing the Identification MDI. Note that the mandatory and optional part of the identification message are not demarcated by DBIDs, hence a device not implementing any of the optional Identification parameters shall fill these bytes with default values as identified in the Command Reference.

A device shall respond to a Set Identification message and update its identification parameters to the values contained in the data payload of the command. It also shall respond to a Get Identification request with a payload containing the version numbers of the ClimateTalk Application, Software Version, Software Revision, and unique serial number at the minimum.

8.0 Error Reporting

The following section defines how communication, data and other errors are reported to the application.

Applications shall handle receiving any of the error messages detailed in this section, and shall format responses to errors it detects as described in this section.

8.1 Invalid Message

If the coordinator does not understand a message sent to it due to an unknown message type or missing or wrong information in the payload, the coordinator will return the message with the bits in the data payload inverted.

8.2 Unknown Application Message Type

If any application receives a message with a value in the message type field that it does not understand, it shall return the message with the original data payload exclusive OR-end with 0x55. This will enable the requesting device to recognize that the message type in the original request was not understood by the application.

8.3 Unknown Application Payload

If any device receives a valid message with a valid message type but a payload that it does not understand, the device will return the message with a 0xA5 appended to the end of the message payload. If the request data payload was 240 bytes, then first 239 bytes will be echoed back and the 240th byte will be replaced with 0xA5 to identify that the request was decoded correctly, but the application is unable to comply.

8.4 Subsystem Busy Response

If the application receives a valid request but is busy at the current time and unable to comply, the application shall notify the requesting application by appending a 0x15 to the request payload and returning it to the requesting application. If the request data payload was 240 bytes, then the first 239 bytes will be echoed back and the 240th byte shall be replaced with 0x15 indicating subsystem/device busy.

8.5 Subsystem Unresponsive

A device application may request anything from any other device on the network. In order to adequately allow for turnaround time in a busy network, the application shall wait a period of time for a response back before concluding that the attempt failed. The turnaround time is called "attempt delay". The attempt delay is 30 seconds. The application should attempt communication five times before concluding that the device queried is in a failure mode of operation. Even after such a conclusion is reached, the attempts must be repeated as long as the attempted transaction is still required.

9.0 User Menus

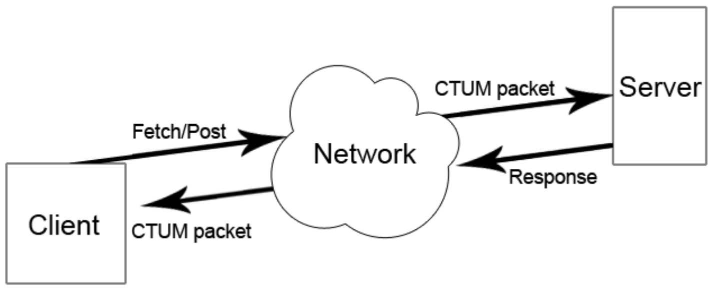
9.1 Support Requirements

Support for User Menu functionality is optional unless otherwise stated by a profile.

9.2 Overview

ClimateTalk User Menus allow communication at the ClimateTalk application layer by sending ClimateTalk User Menu (CTUM) packets through a ClimateTalk Network. This section describes how a device may use user menus to display information to a user.

Figure 4 - ClimateTalk User Menu Block Diagram



A client is any interface that initiates access to information following the ClimateTalk protocol. A server is a device that responds to client requests. A client initiates communication with the server by sending a Fetch or Post at will. The server responds to the client per the CTUM rules. All information must either be Fetched or Posted within one transaction, or within one element of information. These elements may be referred to as CTUM packets.

9.3 System Elements

The ClimateTalk User Menu System is comprised of four elements: a client interface, a CT device node, Fetch methods and Post methods. The CTUM Web Page content and tag usage is derived from the HTML 2.0 web page tag structure. CTUM utilizes a restricted and compressed strategy of tag usage that is structured to provide up to 100 User Menu fields per device that are accessible as 10 Main Menus (maximum) and 10 Sub Menus (maximum) under each Main Menu.

CTUM Tags have been designed for optimal compression and are shown through examples within this document regarding structural usage and recommended CTUM Web Page design.

Additional “Embedded Tags” (eTags) have been designed using supplemental and reserved CTUM eTags that allow functionality beyond the basic HTML screen formatting.

CTUM Main Menus and Submenus can display ASCII alpha-numeric characters as well as basic screen formatting tags that can be decoded in versatile ways by the CTUM client (display device). CTUM Main Menus are referred to as Level 1 while Sub Menus are Level 2.

The menu levels are defined as follows:

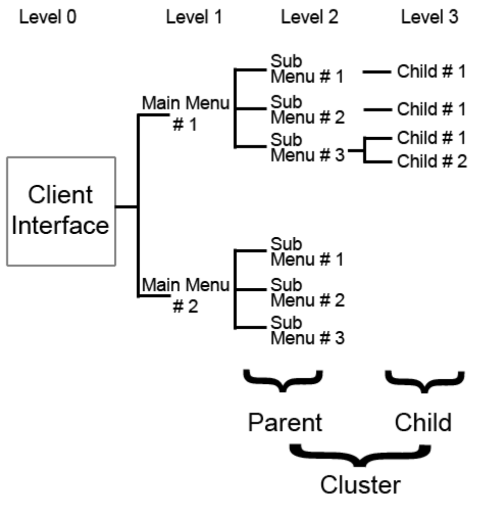
Level 0 – Subsystem / Device
Level 1 – Main Menu
Level 2 – Sub Menu
Level 3 – Child/Children

CTUM Sub Menus can contain dynamic variables as strings, numerical values and selectable fields. Selectable fields can be used to edit fields of data similar to a drop-down menu in HTML 2.0.

9.4 User Menu Organization

This section describes methods for Fetching and Posting CTUM User Menu content from a server to a client. A CTUM User Menu on a device may have up to two levels of navigation. Any option on the second level will have at least one value associated with it. If "selectable", this second level option may allow its value to be selected from a range of specified options (or Children). A second level User Menu item along with all possible values is referred to as a Cluster. Thus, a Cluster will include the parent Sub Menu, the currently assigned value and all other possible values. It is possible to design a Cluster that has a Parent without any children but all children must have a parent.

Figure 5 - User Menu Hierarchy



9.5 User Menu Interface

The CTUM Client interface will allow access to User Menus from all devices in ClimateTalk.

The device displaying the interface will provide a top-level menu display where an individual device's User Menu may be selected. Once the User Menu has been selected, the device will Fetch all Level 1 User Menu information from a specified device node. (A Wildcard Fetch of Level 1 information is used in this situation. Refer to User Menu Fetches for more information.)

9.5.1 Display Devices

Display Devices are devices capable of implementing and displaying user menus. Examples are thermostats, zone controllers and zone thermostats. The implementation of user menus may be required or optional for devices. Devices required to implement user menus are defined in each profile.

9.6 Client Interface

A client is any interface that initiates access to information following the ClimateTalk protocol. A client initiates communication with a server by sending a Fetch or Post at will.

9.6.1 Communication Rules

Maximum 230 bytes of User Menu CTUM Web Page information can be Fetched or Posted at one time.

A six byte preamble/header precedes the CTUM Web Page User Menu content.

Additionally there are four reserved bytes within the standard 240 byte CTUM packet payload.

9.6.2 User Menu Fetches

Each CTUM Fetch instructs the Server what to deliver as the response to the Client's request. The type of fetch is encoded within the CTUM Fetch Payload Header. The header is formatted as follows (where " + # " represents the header offset):

Table 8 - CTUM User Menu Fetch

Byte (from Start of CT Packet Payload Data)	**CTUM Packet Payload Header**	**Values**	**Comments**
+0	Menu File #	0x01	
+1	Main Menu #	0x00 – 0x0A	
+2	Sub Menu #	0x00 – 0x0A	
+3	Fetch Offset in Bytes	MSB	
+4	Fetch Offset in Bytes	LSB	
+5	# of bytes in Fragment Requested		

The above bytes are defined as follows:

- The Menu File number is restricted to 1 (0x01) since only one menu file is currently allowed. Other values may be enabled in future revisions.
- The Main Menu number denotes the index of the Level 1 Menu from which content is to be retrieved. A value of zero (0x00) indicates a Wildcard Fetch of all Level 1 items. Level 1 Menu items are indicated by a value ranging from 1 (0x01) to 10 (0x0A), with a maximum limitation of ten Main Menus.
- Each Main Menu may contain up to ten Level 2 Menu items, indicated by a Sub Menu number ranging from 1 (0x01) to 10 (0x0A). A Sub Menu value of 0 (0x00) indicates a Wildcard Fetch of all Sub Menus within the specified Main Menu.
- The Fetch Offset specifies the number of bytes to index into when retrieving information corresponding to the above items. An offset of zero (0x00) indicates all bytes are retrieved starting from the beginning. An offset of 10 (0x0A) indicates the first ten bytes are to be skipped and the remaining bytes are to be delivered in the response.
- The number of bytes in the fragment requested limits the total number of bytes that will be returned in the response message. This number shall include the size of the CTUM packet payload header including reserved bytes. The maximum value identified inside # of bytes Requested shall be <= 240, 0xF0. Some devices may not be able to process large amounts of CTUM data and may need to make multiple User Menu Fetch requests to Fetch all User Menu information. In this case the MSB and LSB offset parameters (offsets +3 and +4) in the CTUM Packet Payload Header are used to Fetch a CT User Menu in chunks.

For details on the format of User Menu Fetch request and response messages, please consult the Command Reference. The different types of User Menu Fetches performed by varying the values for these parameters are discussed in more depth below.

9.6.2.1 Wildcard Fetch of Level 1

This method allows the User Menu Interface to Fetch all Level 1 Main Menus.

Ex. CTUM packet header: <01><00><00><00><00><F0>

The specification of 0x00 for *Main Menu #* implies a “wildcard”, or the Fetching of all Main Menu items.

9.6.2.2 Fetch of Specific Level 1

This method allows the User Menu Interface to Fetch a specific Level 1 Main Menu and all Level 2 Sub Menus within the specified Main Menu, as well as selected Child values in each Sub Menu.

Ex. CTUM packet header: <01><06><00><00><00><F0>

In this example, the client is requesting User Menu information for Main Menu # 6.

9.6.2.3 Fetch of Specific Cluster

This method allows the User Menu Interface to Fetch a specific Sub Menu and all Children.

Ex. CTUM packet header: <01><06><02><00><00><F0>

In this example, the client is requesting the Sub Menu # 2 Cluster (within Main Menu # 6)

9.6.2.4 Fetch of Entire Web Page

(This Fetch mode is not supported in this version of ClimateTalk, but is reserved for future support.)

This method allows the User Menu Interface to Fetch an entire Web Page.

Ex. CTUM packet header: <01><FF><FF><XX><XX><F0>

In this example, <XX><XX> represents the byte offset to begin the current fetch.

9.6.3 User Menu Update

Each CTUM Update instructs the Server what to update in the User Menu. This request is more than a value update, this is a selection operation allowing a selection of a specific option for which the application may initiate specific operations. The details of what is to be updated are provided as a part of the CTUM Update Packet Payload Header. The header is formatted as follows (where "+ #" is the header offset):

Table 9 - CTUM User Menu Update

Byte (from Start of CT Packet Payload Data)	**CTUM Packet Payload Header**	**Values**	**Comments**
+0	Menu File #	0x01	
+1	Main Menu #	0x00 – 0x0A	
+2	Sub Menu #	0x00 – 0x0A	
+3	File Security Code	0x55	
+4	Updated value to write	MSB	Updated value is equal to the selected option (or Level 3 Child Number) beginning at 1 (0x01)
+5	Updated value to write	LSB	
+6	File Security Code	0xAA	

The above bytes are defined as follows:

- The Menu File number is restricted to 1 (0x01) since only one menu file is currently allowed. Other values may be enabled in future revisions.

- The Main Menu number denotes the index of the Level 1 Menu from which content is to be retrieved. A value of zero (0x00) indicates a Wildcard Fetch of all Level 1 items. Level 1 Menu items are indicated by a value ranging from 1 (0x01) to 10 (0x0A), with a maximum limitation of ten Main Menus.
- Each Main Menu may contain up to ten Level 2 Menu items, indicated by a Sub Menu number ranging from 1 (0x01) to 10 (0x0A). A Sub Menu value of 0 (0x00) indicates a Wildcard Fetch of all Sub Menus within the specified Main Menu.
- The File Security Codes (offsets +3 and +6) are used to verify the intention of the User Menu Update.
- The Updated Value to Write (LSB and MSB) contains the index of the Level 3 Child within the Parent Level 2 Sub Menu to be marked as the new selected value. For example, if the sixth Child of a Sub Menu item is selected, the new selected value would become 6.

For details on a Server's response to a User Menu Update, please consult the Command Reference Specification and applicable API Reference.

9.6.4 Refresh Tags

Refresh tags encoded in the CT User Menu tell the client if and how frequently the page should be re-fetched from the server. CTUM Level 1 contains the Refresh Tag code followed by a Data Tag indicating the period of the refresh in seconds.

A refresh value of zero indicated by a data tag of D0 would indicate a request to the client to attempt to refresh the user menu as fast as possible. A refresh value of one, indicated by a data tag of D1 indicates that the client must attempt to refresh once every second. The slowest refresh rate would be every 15 seconds, indicated by DF. For example, "<C5><D3>" would indicate that the User Menu should be refreshed every 3 seconds.

9.7 Server Interface

A server is a device that responds to client requests. The server responds to the client per the CTUM rules.

9.7.1 Communication Rules

Maximum 230 bytes of User Menu information can be Fetched or Posted at one time.

9.7.2 Response to Client Fetch

The header is formatted as follows (where " + # " is the header offset):

Table 10 - CTUM Response to Client Fetch

Byte (from Start of CT Packet Payload Data)	**CTUM Packet Payload Header**	**Values**	**Comments**
+0	Menu File #	0x01	
+1	Main Menu #	0x00 – 0x0A	

+2	Sub Menu #	0x00 – 0x0A	
+3	Fetch Offset in Bytes	MSB	
+4	Fetch Offset in Bytes	LSB	
+5	# of bytes in Fragment Requested		
+6	File Fragment		Max number of bytes requested

9.7.3 Response to Client Post

The header is formatted as follows (where " + # " is the header offset):

Table 11 - CTUM Response to Client Post

Byte (from Start of CT Packet Payload Data)	**CTUM Packet Payload Header**	**Values**	**Comments**
+0	Menu File #	0x01	
+1	Main Menu #	0x00 – 0x0A	
+2	Sub Menu #	0x00 – 0x0A	
+3	File Security Code	0x55	
+4	Updated value to write	MSB	Updated value is equal to the selected option (or Level 3 Child Number) beginning at 1 (0x01)
+5	Updated value to write	LSB	
+6	File Security Code	0xAA	
+7	Status	0x06	If written successfully

9.7.4 Dynamic Field Text Replacement

Special tags have been developed to allow embedding data code values as DTAGs. Data Tags (DTAGs) are used for special functions as OpCodes. Dynamic variable fields are used by a CTUM server as tagged markers within the CTUM Web Page design that allows the CTUM server to replace fields of data with relevant ASCII display data.

The usage of DTAGs allows a Cluster to contain up to 16 dynamic variables per Cluster. The DTAGs are numbered as DTAG_D0 through DTAG_DF and must be used once and only once per Cluster. Each dynamic variable in CTUM version 1.00 must be treated as a finite dynamic replacement of CTUM characters as fixed length strings and/or fixed length numerical data in ASCII format. When a CTUM server responds to a fetch where the response will include a dynamic field text replacement, the server uses the Cluster information and the DTAG to select the text to be used as a replacement.

All DTAGs that are used for dynamic field text replacement should be stripped from the CTUM Web Page and should never be sent to the CTUM client. The dynamic variable replacement feature is merely using the DTAG as a unique OpCode (per Cluster) in order to allow the CTUM server to easily replace fields of ASCII text with dynamic and relevant ASCII characters of identical length.

9.7.5 Stealth Mode Feature

Each CTUM server can optionally and dynamically restrict the delivery of Level 1 and/or Level 2 fields to a client. This concept allows a CTUM server to hide certain information from a CTUM client.

Any Level 1 or Level 2 field that is desired to be restricted will be invisible to the CTUM client where evidence of these fields is NOT sent to the CT Network. Any field of data that falls into this category must be restricted in its entirety. These fields are considered to be "Stealthed" from the CTUM server perspective.

It is recommended that any Stealthed field should be at or near the end of its Level. For example, if Main Menu #10 is desired to be Stealthed then the CTUM client will see only Main Menu #1 through Main Menu #9, which occurs when the CTUM server declares Main Menu #10 to be Stealthed.

Alternately, if Main Menu #9 is designed to be Stealthed, then the CTUM server would additionally Stealth (or hide content from the client) for any higher numbers of that Level. In this example, the CTUM server would hide the content of Main Menu #9 and Main Menu #10.

Therefore, the Stealthed field that a CTUM server has declared to be hidden from the CTUM client should be the first of its level if it is to be hidden from the CTUM client.

The primary reason for the additional restriction of Stealthing within its Level is that "Higher Menu Number Fields" may contain "selectable" or editable fields. The CTUM system must maintain a contiguous and consistent placement of Main Menu (Level 1) and Sub Menu (Level 2) field numbers that are referenced as numbers 1 through 10.

The best CTUM Stealth design will always strive to have any potential Stealth fields located at or near the last Main Menu or Sub Menu entry possible.

10.0 CT User Menu Tags

This section covers CTUM Tags, CTUM Structure, Control Rules, and Examples.

10.1 CTUM Tag Overview

This section correlates CTUM Tags to CTUM Packet Structure and briefly discusses the transcoding to HTML 2.0.

The Fetched chunks of data are strategically extracted from the comprehensive CTUM Web Page file or content. Various methods of access are supported in CTUM version 1.00 to allow Fetching of Main Menu names, Sub Menu content for a specific Sub Menu, Cluster content for a specific Cluster, and Posting updates to specific Clusters.

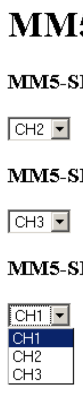
CTUM Tags have evolved from HTML 2.0. They are used as mark-up to format Fetched CT User Menu information.

Ex. CT User Menu – Web Page:

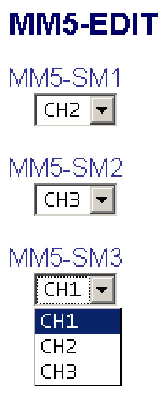
Below are examples of how CT User Menus are represented in HTML. Although there are many ways to visually represent a web page, the following are two examples of web page display: "Basic" and "Expanded". CTUM Basic has evolved directly from HTML 2.0. CTUM Expanded has additional styling to enhance readability. Both examples below were directly transcoded from a CT User Menu originally presented in CTUM format.

Figure 6 - ClimateTalk User Menu viewed as Webpage

Example of CTUM "Basic" HTML:



Example of CTUM "Expanded" HTML:



Ex. CT User Menu – Hierarchy structure:

The example below represents the hierarchal structure of a CT User Menu.

Hierarchal Structure:	Structure based on example above (CTUM as web page):
Main Menu # 5	"MM5-EDIT"
Sub Menu # 1	"MM5-SM1"
Child # 1	"CH1"
Child # 2	**"CH2"**
Child # 3	"CH3"
Sub Menu # 2	"MM5-SM2"
Child # 1	"CH1"
Child # 2	"CH2"
Child # 3	**"CH3"**
Sub Menu # 3	"MM5-SM3"
Child # 1	**"CH1"**
Child # 2	"CH2"
Child # 3	"CH3"

Ex. CT User Menu – Header file:

The example below is a CT User Menu represented as a CTUM Header file, and is the same User Menu as the initial CT User Menu web page example.

```
CTUM_P "MM5-EDIT" CTUM_P_

        CTUM_SELECT  CTUM_EDIT "MM5-SM1"
                CTUM_NORMAL "CH1"
                                CTUM_SEL_NORMAL "CH2"
                                        CTUM_NORMAL "CH3"
        CTUM_SELECT_

        CTUM_SELECT  CTUM_EDIT "MM5-SM2"
                        CTUM_NORMAL "CH1"
                CTUM_NORMAL "CH2"
                                CTUM_SEL_NORMAL "CH3"
        CTUM_SELECT_

        CTUM_SELECT  CTUM_EDIT "MM5-SM3"
                        CTUM_SEL_NORMAL "CH1"
                CTUM_NORMAL "CH2"
                CTUM_NORMAL "CH3"
        CTUM_SELECT_
```

NOTE: The complete CTUM Header file for these examples can be found at the end of this document. CTUM Header files can be compiled directly in ANSI-C.

The equivalent HTML Code for the Basic HTML example above can be transcoded as follows:

This example is the HTML code for a CT User Menu file formatted as a CTUM Basic HTML web page, and contains the same content as the User Menu as the initial CT User Menu web page example. The HTML code shown below is the result after transcoding a CTUM Basic HTML web page. This is only an example of a transcoding format and many display formats are possible.

The following items are synonymous in the CTUM examples within this document section:

1. CT User Menu represented in the Header file (example above)
2. CT User Menu represented in the CTUM Packet Payload (example below)
3. CT User Menu represented as Basic HTML
4. CT User Menu represented as Expanded HTML
5. CT User Menu represented in any other textual format desired

Ex. Transcoded Basic HTML derived from CTUM Header File:

```
<HTML>

        <BODY>

        <h1>MM5-EDIT</h1>

                <h3>MM5-SM1</h3>
                        <form><select name='MM5SM1' onchange='submit()'>
                                <option value='1'>CH1</option>
                                <option selected='selected' value='2'>CH2</option>
                                <option value='3'>CH3</option>
                        </select></form>

                <h3>MM5-SM2</h3>
                        <form><select name='MM5SM2' onchange='submit()'>
                                <option value='1'>CH1</option>
                                <option value='2'>CH2</option>
                                <option selected='selected' value='3'>CH3</option>
                        </select></form>

                <h3>MM5-SM3</h3>
                        <form><select name='MM5SM3' onchange='submit()'>
                                <option selected='selected' value='1'>CH1</option>
                                <option value='2'>CH2</option>
                                <option value='3'>CH3</option>
                        </select></form>

        </BODY>

</HTML>
```

Ex. CT User Menu – Packet Payload

The equivalent Fetch for this example in CTUM (Hex format) is:

CTUM Header (contained in CT Packet Payload [+0 offset]):

```
<01><05><00><00><00><F0>  // This Header Code says Fetch Main Menu # 5.
```

CTUM Fetch Response from Server to Client (immediately after the CTUM Header [+6 offset]):

```
[------CTUM_HEADER-----
]<B6><4D><4D><35><2D><45><44><49><54><B7><C2><E1><4D><4D><35>
<2D><53><4D><31><E0><43><48><31><E8><43><48><32><E0><43><48><33>
<C3><C2><E1><4D><4D>
<35><2D><53><4D><32><E0><43><48><31><E0><43><48><32><E8><43><48>
<33><C3><C2><E1><4D>
<4D><35><2D><53><4D><33><E8><43><48><31><E0><43><48><32><E0><43
><48><33><C3>
```

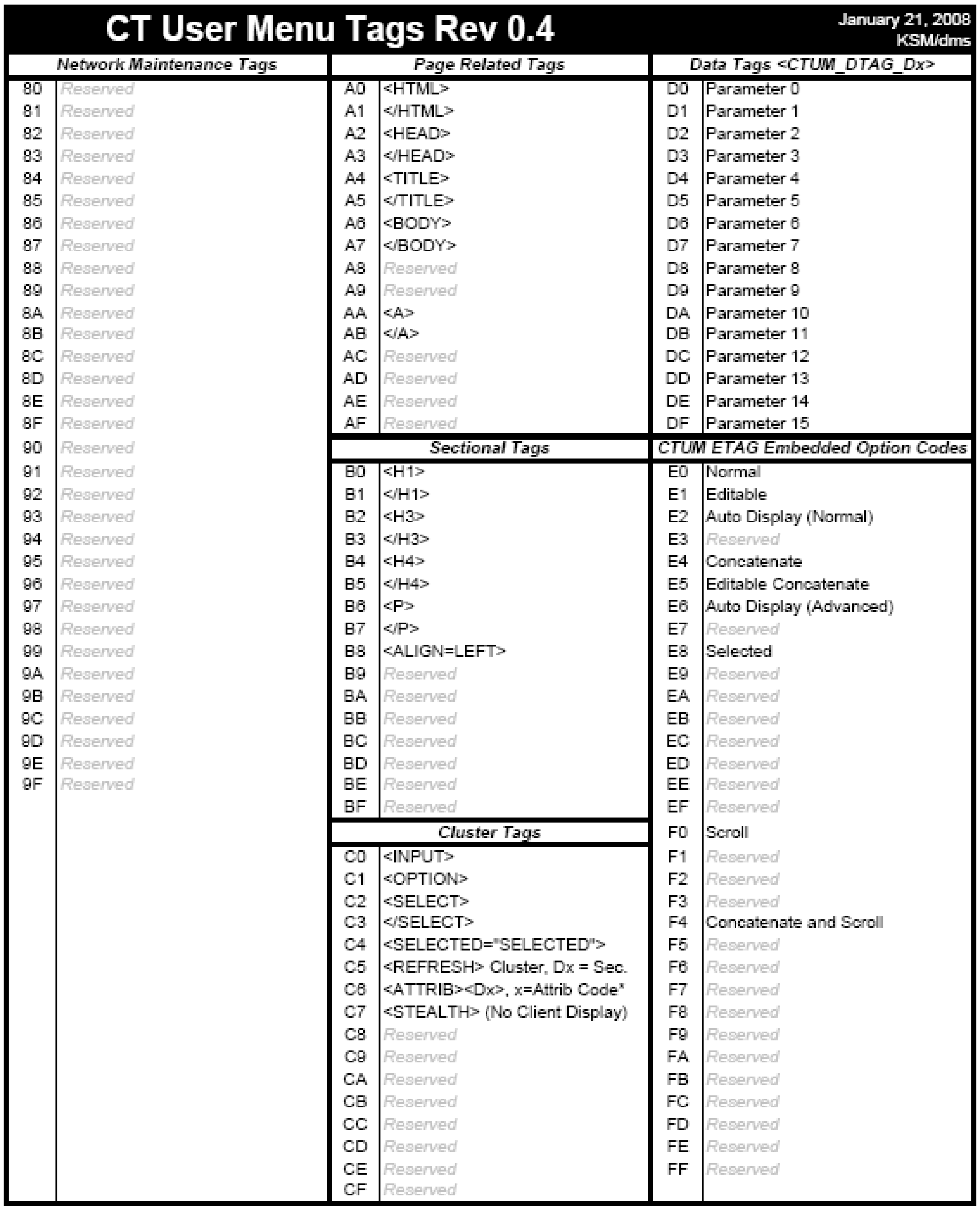
NOTE: Please see the CTUM Tag values decoded below, as shown in hexadecimal format above.

10.1.1CT User Menu Tags

The table below defines all supported CTUM Version 1.00 User Menu Tag values and their names.

All other values not in this table are simple ASCII text.

Figure 7 - ClimateTalk User Menu Tag Reference Table



CT User Menu Tags Rev 0.4

January 21, 2008
KSM/dms

Network Maintenance Tags

80	Reserved
81	Reserved
82	Reserved
83	Reserved
84	Reserved
85	Reserved
86	Reserved
87	Reserved
88	Reserved
89	Reserved
8A	Reserved
8B	Reserved
8C	Reserved
8D	Reserved
8E	Reserved
8F	Reserved
90	Reserved
91	Reserved
92	Reserved
93	Reserved
94	Reserved
95	Reserved
96	Reserved
97	Reserved
98	Reserved
99	Reserved
9A	Reserved
9B	Reserved
9C	Reserved
9D	Reserved
9E	Reserved
9F	Reserved

Page Related Tags

A0	<HTML>
A1	</HTML>
A2	<HEAD>
A3	</HEAD>
A4	<TITLE>
A5	</TITLE>
A6	<BODY>
A7	</BODY>
A8	Reserved
A9	Reserved
AA	<A>
AB	</A>
AC	Reserved
AD	Reserved
AE	Reserved
AF	Reserved

Sectional Tags

B0	<H1>
B1	</H1>
B2	<H3>
B3	</H3>
B4	<H4>
B5	</H4>
B6	<P>
B7	</P>
B8	<ALIGN=LEFT>
B9	Reserved
BA	Reserved
BB	Reserved
BC	Reserved
BD	Reserved
BE	Reserved
BF	Reserved

Cluster Tags

C0	<INPUT>
C1	<OPTION>
C2	<SELECT>
C3	</SELECT>
C4	<SELECTED="SELECTED">
C5	<REFRESH> Cluster, Dx = Sec.
C6	<ATTRIB><Dx>, x=Attrib Code*
C7	<STEALTH> (No Client Display)
C8	Reserved
C9	Reserved
CA	Reserved
CB	Reserved
CC	Reserved
CD	Reserved
CE	Reserved
CF	Reserved

Data Tags <CTUM_DTAG_Dx>

D0	Parameter 0
D1	Parameter 1
D2	Parameter 2
D3	Parameter 3
D4	Parameter 4
D5	Parameter 5
D6	Parameter 6
D7	Parameter 7
D8	Parameter 8
D9	Parameter 9
DA	Parameter 10
DB	Parameter 11
DC	Parameter 12
DD	Parameter 13
DE	Parameter 14
DF	Parameter 15

CTUM ETAG Embedded Option Codes

E0	Normal
E1	Editable
E2	Auto Display (Normal)
E3	Reserved
E4	Concatenate
E5	Editable Concatenate
E6	Auto Display (Advanced)
E7	Reserved
E8	Selected
E9	Reserved
EA	Reserved
EB	Reserved
EC	Reserved
ED	Reserved
EE	Reserved
EF	Reserved
F0	Scroll
F1	Reserved
F2	Reserved
F3	Reserved
F4	Concatenate and Scroll
F5	Reserved
F6	Reserved
F7	Reserved
F8	Reserved
F9	Reserved
FA	Reserved
FB	Reserved
FC	Reserved
FD	Reserved
FE	Reserved
FF	Reserved

* Client Attribute Code Bit Encoding = <Dx>, where x = D3-Year, D2-Month, D1-Days, D0-Hours (Segments Lit =High).

10.1.2CTUM Structure as Web Page

A CTUM file is a ClimateTalk User Menu web page per the standards release of CTUM version 1.00.

CTUM tags supported in CTUM version 1.00 are outlined in the following rules and are shown with their appropriate usage in Annex A - Sample CTUM Header File. This file is intended to be a guideline for all supported Main Menu and Sub Menu structures. The full CTUM Tag outline can be found in the previous section of this document.

The beginning of a CTUM file is identified by *CTUM_HTML* (CTUM open HTML tag) followed by the corresponding *CTUM_BODY* (CTUM open body tag).

The end of a CTUM file is identified by *CTUM_BODY_* (CTUM close body tag) followed by the corresponding *CTUM_HTML_* (CTUM close HTML tag).

10.1.2.1 Main Menus (Level 1)

A CTUM web page file can contain between 1 and 10 Main Menus. Main Menus exist within the body of a CTUM file, between *CTUM_BODY* and *CTUM_BODY_*.

The beginning of each Main Menu is identified by *CTUM_P* (CTUM open paragraph tag).

The end of each Main Menu is identified by *CTUM_P_* (CTUM close paragraph tag).

Supplemental information also resides within the definition of a Main Menu. Optionally, if the Main Menu is to be refreshed, *CTUM_REFRESH* (CTUM refresh tag) must immediately follow *CTUM_P*. A tag indicating magnitude must immediately follow *CTUM_REFRESH*, identified by *CTUM_DTAG_D(0xA-F)* (CTUM open DTAG, followed by a hexadecimal value indicating the rate in seconds). The name of the Main Menu, also residing in the definition, is represented as a text field and is encapsulated in double quotations.

For example, in Annex A - Sample CTUM Header File the line containing "*CTUM_P CTUM_REFRESH CTUM_DTAG_D5 "MM1-NEW1" CTUM_P_*" identifies Main Menu one named MM1-NEW1 and refreshes at a rate of every 5 seconds.

There is a minimum limitation of 1 Main Menu and a maximum limitation of 10 Main Menus per CTUM file. In order to count the number of Main Menus within the CTUM file, count the number of CTUM open paragraph tags until *CTUM_BODY_* is found. If the CTUM close body tag is not present, then count until the end of the fetch packet is reached.

10.1.2.2 Sub Menus (Level 2)

Similar to Main Menus, Sub Menus exist within the body of a CTUM file. Sub Menus follow the definition of a Main Menu.

After first identifying a Main Menu, the Sub Menu can then be identified by *CTUM_SELECT* (CTUM open select tag).

The end of a Sub Menu is identified by *CTUM_SELECT_* (CTUM close select tag).

Supplemental information can also reside within the definition of a Sub Menu. The name of the Sub Menu is represented as a text field and is encapsulated in double quotations. Children can optionally be contained within the Sub Menu definition.

For example, in Annex A - Sample CTUM Header File the line containing "*CTUM_SELECT CTUM_NORMAL "MM1-SM2""* identifies the beginning of Sub Menu 2 named MM1-SM2 within MM #1. Supplemental tags, known as eTags, can optionally be included following the text field of a Sub Menu. This is how the definition of Children is established.

10.1.2.3 Children (Level 3)

Similar to Main Menus and Sub Menus, Children exist within the body of a CTUM file. Children exist within the definition of Sub Menus.

Children are identified by eTags: *CTUM_NORMAL* (CTUM normal option tag), *CTUM_SEL_NORMAL* (CTUM selected normal option tag), *CTUM_AUTO* (CTUM auto option tag), and *CTUM_AUTO_ADV* (CTUM auto-advanced option tag), are encapsulated within a Sub Menu (CTUM open and close select tags), and are a part of the cluster.

The number of children is determined by the number of eTags following the name of the Sub Menu and before the CTUM close select tag.

For example, in Annex A - Sample CTUM Header File the line containing "*CTUM_NORMAL "654""* identifies the first (and only) child within MM #1 SM #3.

10.1.2.4 Clusters

A cluster is a combination of a specific Sub Menu and all associated Children, or the content defined within CTUM open and close select tags.

The following is an example of a cluster, found in Annex A - Sample CTUM Header File: *"CTUM_SELECT CTUM_AUTO "MM1-SM1" CTUM_NORMAL "MM1SM1-CH1" CTUM_SELECT_"*.

10.1.3 Embedded Option Codes, Expanded

Additional "Embedded Tags" (eTags) have been designed using supplemental and reserved CTUM eTags that allow functionality beyond the basic HTML screen formatting as detailed below:

- Normal <0xE0> – No special action taken for this item.
- Selectable <0xE1> – Allows modification of a Sub Menu. If a Sub Menu is selectable, then a user may modify the menu by choosing a different Child than what is selected by default.
- Auto Display (Normal) <0xE2> – Level 2 text will be displayed, then corresponding Level 3 text will be displayed.
- Concatenate <0xE4> – This item will be concatenated with its selection Level 3 Child using an underscore "_" separator.
- Selectable Concatenate <0xE5> – This item is selectable ; it will be concatenated with its selection level items using an underscore "_" separator.
- Auto Display (Advanced) <0xE6> – Level 3 primary text will be displayed initially, then alternate Level 3 text will be displayed.
- Selected <0xE8> – Default selection for a selectable Sub Menu.
- Scroll – This item will be scrolled by the client per the rules of the client.

- Concatenate and Scroll – This item will be concatenated with its selection Level 3 Child and scrolled per the rules of the client.

10.1.4 Definition of CTUM Tag Names

The following are the ANSI-C tokens shown in hexadecimal and are used in examples shown throughout this document.

Figure 8 - CTUM Tag Conversion Table

Hex		CTUM Tag		Description	Hex		CTUM Tag		Description
A0	=	CTUM_HTML	=	Open HTML	D0	=	CTUM_DTAG_D0	=	DataTag (D0)
A1	=	CTUM_HTML_	=	Close HTML	D1	=	CTUM_DTAG_D1	=	DataTag (D1)
A6	=	CTUM_BODY	=	Open Body	D2	=	CTUM_DTAG_D2	=	DataTag (D2)
A7	=	CTUM_BODY_	=	Close Body	D3	=	CTUM_DTAG_D3	=	DataTag (D3)
					D4	=	CTUM_DTAG_D4	=	DataTag (D4)
B6	=	CTUM_P	=	Open Paragraph	D5	=	CTUM_DTAG_D5	=	DataTag (D5)
B7	=	CTUM_P_	=	Close Paragraph	D6	=	CTUM_DTAG_D6	=	DataTag (D6)
					D7	=	CTUM_DTAG_D7	=	DataTag (D7)
C2	=	CTUM_SELECT	=	Open Select	D8	=	CTUM_DTAG_D8	=	DataTag (D8)
C3	=	CTUM_SELECT_	=	Close Select	D9	=	CTUM_DTAG_D9	=	DataTag (D9)
C5	=	CTUM_REFRESH	=	Refresh	DA	=	CTUM_DTAG_DA	=	DataTag (D10)
					DB	=	CTUM_DTAG_DB	=	DataTag (D11)
					DC	=	CTUM_DTAG_DC	=	DataTag (D12)
					DD	=	CTUM_DTAG_DD	=	DataTag (D13)
					DE	=	CTUM_DTAG_DE	=	DataTag (D14)
					DF	=	CTUM_DTAG_DF	=	DataTag (D15)
					E0	=	CTUM_NORMAL	=	Normal Field
					E1	=	CTUM_EDIT	=	Editable Field
					E2	=	CTUM_AUTO	=	Auto Field
					E4	=	CTUM_CONCAT	=	Concatenate Field
					E5	=	CTUM_CONCAT_EDIT	=	Editable Concatenate Field
					E6	=	CTUM_AUTO_ADV	=	Auto Advanced Field
					E8	=	CTUM_SEL_NORMAL	=	Selected Field

10.2 Control Rules

ClimateTalk User Menu control rules outline how CTUM Tags affect Selection Level items.

10.2.1 Main Menu Control Rules

There are no options for this menu.

10.2.2 Sub Menu Control Rules

There are three types of Selection Level items that may change the behavior of the Sub Menu Level:

Sub Menu Types:

1. Normal
   - option code = 0x00 = <E0>
2. Auto Display (Normal)
   - option code = 0x02 = <E2>
   - Auto Display shows the Selection Level item for the current Sub Menu item until a button is pressed
3. Selectable Concatenate
   - option code = 0x04 = <E4>
   - Sub Menu item auto displays default Selection concatenated with the Sub Menu item.

10.2.3 Selection Control Rules

There are three types of selection level items, only selectable and multiple non-selectable types function at this level:

Selection Level Types:

1. Normal
   - Option code = 0x00 = <E0>
2. Auto Display (Advanced)
   - Option code = 0x06 = <E6>
   - Auto Display (Advanced) items have a Primary and Alternate text associated with them.
   - On entry, the Primary text is displayed
3. Selected
   - Option code = 0x08 = <E8>
   - Only one selection per Sub Menu can be tagged as selected
   - The selection tagged “selected” will be considered default and will be displayed first.

11.0 Annex A - Sample CTUM Header File

```
//---------------------------------------------------------------------------------------------------------------------------------
// This file contains the actual Data (or Web Page CTmicroTagged HTML Code) that resides on the HP/IFC/AH, etc. Subsystem.
//
// The Web Page is Stored in the Subsystem's APPLICATION Program Memory.
//---------------------------------------------------------------------------------------------------------------------------------

const char * UserMenu1=

CTUM_HTML                // HTML - Begins WEB PAGE (Main Menu / Level I Fetch Payload = 118)

        CTUM_BODY        // Start of the actual CTUM TAGS that define the actual User Menus for this Web Page.

                CTUM_P CTUM_REFRESH CTUM_DTAG_D5 "MM1-NEW1" CTUM_P_                    // LEVEL I - MM # 1 (Fetch Payload = 66)

                        CTUM_SELECT CTUM_AUTO "MM1-SM1"            // LEVEL II - MM # 1 SM #1
                                CTUM_NORMAL "MM1SM1-CH1"    // LEVEL III - MM # 1 SM # 1 CH # 1
                        CTUM_SELECT_

                        CTUM_SELECT CTUM_NORMAL "MM1-SM2"          // LEVEL II - MM # 1 SM #2
                                CTUM_NORMAL "987"           // LEVEL III - MM # 1 SM # 2 CH # 1
                        CTUM_SELECT_

                        CTUM_SELECT CTUM_NORMAL "MM1-SM3"          // LEVEL II - MM # 1 SM #3
                                CTUM_NORMAL "654"           // LEVEL III - MM # 1 SM # 3 CH # 1
                        CTUM_SELECT_


                CTUM_P "MM2-AUTO" CTUM_P_                                      // LEVEL I - MM # 2 (Fetch Payload = 79)

                        CTUM_SELECT CTUM_AUTO "MM2-SM1"            // LEVEL II - MM # 2 SM #1
                                CTUM_NORMAL "MM2SM1-CH1"    // LEVEL III - MM # 2 SM # 1 CH # 1
                        CTUM_SELECT_

                        CTUM_SELECT CTUM_AUTO "MM2-SM2"            // LEVEL II - MM # 2 SM #2
                                CTUM_NORMAL "MM2SM2-CH1"    // LEVEL III - MM # 2 SM # 2 CH # 1
                        CTUM_SELECT_

                        CTUM_SELECT CTUM_AUTO "MM2-SM3"            // LEVEL II - MM # 2 SM #3
                                CTUM_NORMAL "MM2SM3-CH1"    // LEVEL III - MM # 2 SM # 3 CH # 1
                        CTUM_SELECT_
```

```
CTUM_P "MM3-ATOADV" CTUM_P_                                    // LEVEL I - MM # 3 (Fetch Payload = 228)

    CTUM_SELECT CTUM_NORMAL "MM3-SM1"                          // LEVEL II - MM # 3 SM #1
        CTUM_AUTO_ADV "MM3SM1-CH1CHILD 1"   // LEVEL III - MM # 3 SM # 1 CH # 1
        CTUM_AUTO_ADV "MM3SM1-CH2CHILD 2"   // LEVEL III - MM # 3 SM # 1 CH # 2
        CTUM_AUTO_ADV "MM3SM1-CH3CHILD 3"   // LEVEL III - MM # 3 SM # 1 CH # 3
    CTUM_SELECT_

    CTUM_SELECT CTUM_NORMAL "MM3-SM2"                          // LEVEL II - MM # 3 SM #2
        CTUM_AUTO_ADV "MM3SM2-CH1CHILD 1"   // LEVEL III - MM # 3 SM # 2 CH # 1
        CTUM_AUTO_ADV "MM3SM2-CH2CHILD 2"   // LEVEL III - MM # 3 SM # 2 CH # 2
        CTUM_AUTO_ADV "MM3SM2-CH3CHILD 3"   // LEVEL III - MM # 3 SM # 2 CH # 3
    CTUM_SELECT_

    CTUM_SELECT CTUM_NORMAL "MM3-SM3"                          // LEVEL II - MM # 3 SM #3
        CTUM_AUTO_ADV "MM3SM3-CH1CHILD 1"   // LEVEL III - MM # 3 SM # 3 CH # 1
        CTUM_AUTO_ADV "MM3SM3-CH2CHILD 2"   // LEVEL III - MM # 3 SM # 3 CH # 2
        CTUM_AUTO_ADV "MM3SM3-CH3CHILD 3"   // LEVEL III - MM # 3 SM # 3 CH # 3
        CTUM_AUTO_ADV "MM3SM3-CH3CHILD 4"   // LEVEL III - MM # 3 SM # 3 CH # 4
    CTUM_SELECT_


CTUM_P "MM4-C-EDIT" CTUM_P_                                    // LEVEL I - MM # 4 (Fetch Payload = 81)

    CTUM_SELECT CTUM_CONCAT_EDIT "MM4SM1"                      // LEVEL II - MM # 4 SM #1
        CTUM_NORMAL "CH1"               // LEVEL III - MM # 4 SM # 1 CH # 1
        CTUM_SEL_NORMAL "CH2"                   // LEVEL III - MM # 4 SM # 1 CH # 2 - SELECTED
        CTUM_NORMAL "CH3"               // LEVEL III - MM # 4 SM # 1 CH # 3
    CTUM_SELECT_

    CTUM_SELECT CTUM_CONCAT_EDIT "MM4SM2"                      // LEVEL II - MM # 4 SM #2
        CTUM_NORMAL "CH1"               // LEVEL III - MM # 4 SM # 2 CH # 1
        CTUM_NORMAL "CH2"               // LEVEL III - MM # 4 SM # 2 CH # 2
        CTUM_SEL_NORMAL "CH3"                   // LEVEL III - MM # 4 SM # 2 CH # 3 - SELECTED
    CTUM_SELECT_

    CTUM_SELECT CTUM_CONCAT_EDIT "MM4SM3"                      // LEVEL II - MM # 4 SM #3
        CTUM_SEL_NORMAL "CH1"                   // LEVEL III - MM # 4 SM # 3 CH # 1 - SELECTED
        CTUM_NORMAL "CH2"               // LEVEL III - MM # 4 SM # 3 CH # 2
        CTUM_NORMAL "CH3"               // LEVEL III - MM # 4 SM # 3 CH # 3
        CTUM_SELECT_




CTUM_P "MM5-EDIT" CTUM_P_                                    // LEVEL I - MM # 5 (Fetch Payload = 82)

        CTUM_SELECT CTUM_EDIT "MM5-SM1"              // LEVEL II - MM # 5 SM #1
                CTUM_NORMAL "CH1"            // LEVEL III - MM # 5 SM # 1 CH # 1
                CTUM_SEL_NORMAL "CH2"                // LEVEL III - MM # 5 SM # 1 CH # 2 - SELECTED
                CTUM_NORMAL "CH3"            // LEVEL III - MM # 5 SM # 1 CH # 3
        CTUM_SELECT_

        CTUM_SELECT CTUM_EDIT "MM5-SM2"              // LEVEL II - MM # 5 SM #2
                CTUM_NORMAL "CH1"            // LEVEL III - MM # 5 SM # 2 CH # 1
                CTUM_NORMAL "CH2"            // LEVEL III - MM # 5 SM # 2 CH # 2
                CTUM_SEL_NORMAL "CH3"                // LEVEL III - MM # 5 SM # 2 CH # 3 - SELECTED
        CTUM_SELECT_

        CTUM_SELECT CTUM_EDIT "MM5-SM3"              // LEVEL II - MM # 5 SM #3
                CTUM_SEL_NORMAL "CH1"                // LEVEL III - MM # 5 SM # 3 CH # 1 - SELECTED
                CTUM_NORMAL "CH2"            // LEVEL III - MM # 5 SM # 3 CH # 2
                CTUM_NORMAL "CH3"            // LEVEL III - MM # 5 SM # 3 CH # 3
        CTUM_SELECT_


CTUM_P "SETUP" CTUM_P_                                           // LEVEL I - MM # 6 (Fetch Payload = 224)

        CTUM_SELECT CTUM_NORMAL "SM1_NORM"               // LEVEL II - MM # 6 SM #1
                CTUM_NORMAL "NORM1"          // LEVEL III - MM # 6 SM # 1 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_REFRESH CTUM_DTAG_D5  CTUM_AUTO "SM2_AUTO"      // LEVEL II - MM # 6 SM #2
                CTUM_NORMAL "AUTO2"          // LEVEL III - MM # 6 SM # 2 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "AIRTRM"            // LEVEL II - MM # 6 SM #3
                CTUM_NORMAL "-10%"           // LEVEL III - MM # 6 SM # 3 CH # 1
                CTUM_NORMAL "-8%"            // LEVEL III - MM # 6 SM # 3 CH # 2
                CTUM_NORMAL "-6%"            // LEVEL III - MM # 6 SM # 3 CH # 3
                CTUM_NORMAL "-4%"            // LEVEL III - MM # 6 SM # 3 CH # 4
                CTUM_NORMAL "-2%"            // LEVEL III - MM # 6 SM # 3 CH # 5
                CTUM_SEL_NORMAL "0%"         // LEVEL III - MM # 6 SM # 3 CH # 6 - SELECTED
                CTUM_NORMAL "2%"                // LEVEL III - MM # 6 SM # 3 CH # 7
                CTUM_NORMAL "4%"                // LEVEL III - MM # 6 SM # 3 CH # 8
                CTUM_NORMAL "6%"                // LEVEL III - MM # 6 SM # 3 CH # 9
                CTUM_NORMAL "8%"                // LEVEL III - MM # 6 SM # 3 CH # 10
                CTUM_NORMAL "10%"               // LEVEL III - MM # 6 SM # 3 CH # 11
        CTUM_SELECT_

        CTUM_SELECT CTUM_REFRESH  CTUM_DTAG_DA  CTUM_AUTO "SM4_AUTO" // LEVEL II - MM # 6 SM #4
                CTUM_NORMAL "AUTO4"             // LEVEL III - MM # 6 SM # 4 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "HP PRFL"          // LEVEL II - MM # 6 SM #5
                CTUM_SEL_NORMAL "A"             // LEVEL III - MM # 6 SM # 5 CH # 1 - SELECTED
                CTUM_NORMAL "B"                 // LEVEL III - MM # 6 SM # 5 CH # 2
                CTUM_NORMAL "C"                 // LEVEL III - MM # 6 SM # 5 CH # 3
                CTUM_NORMAL "D"                 // LEVEL III - MM # 6 SM # 5 CH # 4
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "CL TRM"                   // LEVEL II - MM # 6 SM #6
                CTUM_NORMAL "-10%"              // LEVEL III - MM # 6 SM # 6 CH # 1
                CTUM_SEL_NORMAL "0%"            // LEVEL III - MM # 6 SM # 6 CH # 2 - SELECTED
                CTUM_NORMAL "10%"               // LEVEL III - MM # 6 SM # 6 CH # 3
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "HT TRM"                   // LEVEL II - MM # 6 SM #7
                CTUM_NORMAL "-15%"              // LEVEL III - MM # 6 SM # 7 CH # 1
                CTUM_NORMAL "0%"                // LEVEL III - MM # 6 SM # 7 CH # 2
                CTUM_SEL_NORMAL "15%"                   // LEVEL III - MM # 6 SM # 7 CH # 3 - SELECTED
        CTUM_SELECT_

        CTUM_SELECT CTUM_EDIT "RESET"                           // LEVEL II - MM # 6 SM #8
                CTUM_NORMAL "NO"                // LEVEL III - MM # 6 SM # 8 CH # 1
                CTUM_NORMAL "YES"               // LEVEL III - MM # 6 SM # 8 CH # 2
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "DEHUM"                    // LEVEL II - MM # 6 SM #9
                CTUM_SEL_NORMAL "ON"            // LEVEL III - MM # 6 SM # 9 CH # 1 - SELECTED
                CTUM_NORMAL "OFF"               // LEVEL III - MM # 6 SM # 9 CH # 2
        CTUM_SELECT_

        CTUM_SELECT CTUM_AUTO "SM10_AUTO"               // LEVEL II - MM # 6 SM #10
                CTUM_NORMAL "AUTO10"            // LEVEL III - MM # 6 SM # 10 CH # 1
        CTUM_SELECT_
```

```
CTUM_P "MAINMENU-7" CTUM_P_                                         // LEVEL I - MM # 7 (Fetch Payload = 236)

        CTUM_SELECT CTUM_NORMAL "FAULT1_XXX"                        // LEVEL II - MM # 7 SM #1
                CTUM_AUTO_ADV "DAYSAGO 1TTEST1"     // LEVEL III - MM # 7 SM # 1 CH # 1
                CTUM_AUTO_ADV "DAYSAGO 2TTEST2"     // LEVEL III - MM # 7 SM # 1 CH # 2
                CTUM_AUTO_ADV "DAYSAGO 3TTEST3"     // LEVEL III - MM # 7 SM # 1 CH # 3
                CTUM_AUTO_ADV "DAYSAGO 4TTEST4"     // LEVEL III - MM # 7 SM # 1 CH # 4
        CTUM_SELECT_

        CTUM_SELECT CTUM_NORMAL "FAULT2_XXX"                        // LEVEL II - MM # 7 SM #2
                CTUM_AUTO_ADV "DAYSAGO A2TEST2A"    // LEVEL III - MM # 7 SM # 2 CH # 1
                CTUM_AUTO_ADV "DAYSAGO B2TEST2B"    // LEVEL III - MM # 7 SM # 2 CH # 2
        CTUM_SELECT_

        CTUM_SELECT CTUM_NORMAL "FAULT3_XXX"                        // LEVEL II - MM # 7 SM #3
                CTUM_AUTO_ADV "DAYSAGO A3TEST3A"    // LEVEL III - MM # 7 SM # 3 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_CONCAT_EDIT "CLR"                  // LEVEL II - MM # 7 SM #4
                CTUM_NORMAL "NO"                    // LEVEL III - MM # 7 SM # 4 CH # 1
                CTUM_NORMAL "YES"                   // LEVEL III - MM # 7 SM # 4 CH # 2
        CTUM_SELECT_

        CTUM_SELECT CTUM_AUTO "MOD NUM A"                   // LEVEL II - MM # 7 SM #5
                CTUM_NORMAL "0123456789"            // LEVEL III - MM # 7 SM # 5 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_AUTO "SER NUM A"                   // LEVEL II - MM # 7 SM #6
                CTUM_NORMAL "543210-ABCDEFGH"               // LEVEL III - MM # 7 SM # 6 CH # 1
        CTUM_SELECT_


CTUM_P "UNIT INFO" CTUM_P_                                          // LEVEL I - MM # 8 (Fetch Payload = 86)

        CTUM_SELECT CTUM_AUTO "MOD NUM B"                   // LEVEL II - MM # 8 SM #1
                CTUM_NORMAL "1234567890ABCDEFG"     // LEVEL III - MM # 8 SM # 1 CH # 1
        CTUM_SELECT_

        CTUM_SELECT CTUM_AUTO "SER NUM B"                   // LEVEL II - MM # 8 SM #2
                CTUM_NORMAL "ABCDEFGHIJKLMNOPQRSTUVWXYZ" // LEVEL III - MM # 8 SM # 2 CH # 1
        CTUM_SELECT_


CTUM_P "JIM NEW P9" CTUM_P_                                         // LEVEL I - MM # 9 (Fetch Payload = 230)
```

```
                CTUM_SELECT CTUM_NORMAL "SINGLE PAR"                    // LEVEL II - MM # 9 SM #1
                CTUM_SELECT_

                CTUM_SELECT CTUM_AUTO "TEST MM9-2"              // LEVEL II - MM # 9 SM #2
                        CTUM_NORMAL "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ"      // LEVEL III - MM # 9 SM # 2 CH # 1
                CTUM_SELECT_

                CTUM_SELECT CTUM_NORMAL "EMPTYCHILD"                    // LEVEL II - MM # 9 SM #3
                CTUM_SELECT_


                CTUM_SELECT CTUM_AUTO "TEST MM9-3"              // LEVEL II - MM # 9 SM #4
                        CTUM_NORMAL "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ"      // LEVEL III - MM # 9 SM # 4 CH # 1
                CTUM_SELECT_



        CTUM_P  CTUM_REFRESH  CTUM_DTAG_D0 "JIMNEW P10" CTUM_P_                 // LEVEL I - MM # 10 (Fetch Payload = 58)

                CTUM_SELECT CTUM_NORMAL "SINGLE P-1"                    // LEVEL II - MM # 10 SM #1
                CTUM_SELECT_

                CTUM_SELECT CTUM_NORMAL "SINGLE P-2"                    // LEVEL II - MM # 10 SM #2
                CTUM_SELECT_

                CTUM_SELECT CTUM_NORMAL "SINGLE P-3"                    // LEVEL II - MM # 10 SM #3
                CTUM_SELECT_

        CTUM_BODY_

    CTUM_HTML_
    ; // This is the End of the UserMenu1.

// END OF FILE
//-----------------------------------------------------------------------------------------------------------------------------------
```

12.0 Annex B – Sample Shared Data Structure

The data area of Shared Data may be arranged in logical groupings and identified with a one byte "field tag" representing the data that follows. A Shared Data record may thus be organized into a number of fields each of which contains a field tag, a field length, and the data within the field. An example of one Shared Data sector organized into fields is shown below.

Figure 9- Shared Data Example

----------start of ***Factory Defined*** Shared Data

field tag = Control information:
    Model number
    etc

field tag = System parameters not modified via user menus
    Indoor blower horsepower
    Indoor blower maximum CFM
    etc

field tag = Indoor blower motor information:
    Motor manufacturer id0
    Number of coefficients used by motor manufacturer id0
    Motor manufacturer id1
    Number of coefficients used by motor manufacturer id1
    etc

field tag = Default values for user menu modified options

----------end of ***factory defined*** Shared Data

----------start of ***User Menu Modified*** Shared Data

field tag = User menu modified options

----------end of ***User Menu Modified*** Shared Data

Table 12 - Shared Data Organization represents the organization of an example Shared Data set.

Within the shared data set there are factory-defined and user-modifiable field. Checksums provided for the different sections ensure integrity of the data is maintained. There are also elements added into the shared data set to allow for forward and backward compatibility of the shared data programming files. The legend that describes the purpose of the shared data elements is provided in Table 13 - Shared Data Legend.

Table 12 - Shared Data Organization

Shared Data Element	1	2	3	4	5
Total number of bytes in Shared Data including this byte and checksums	X		X		
Control ID (2 Bytes)	X		X		
Manufacturer ID (2 Bytes)	X		X		
Device node type	X		X		
# of field IDs to be extracted from Shared Data		X	X		
field ID 0		X	X		
bytes in field ID 0 = n		X	X		
field ID 0 data [0]			X		
…			X		
field ID 0 data [n-1]			X		
field ID 1		X	X		
bytes in field ID 1 = m		X	X		
field ID 1 data [0]			X		
…			X		
field ID 1 data [m-1]			X		
field ID 2		X	X		
bytes in field ID 2 = k		x	X		
field ID 2 data [0]			X		
…			X		
field ID 2 data [k-1]			X		
field ID = user menu data: default values		X	X		
bytes in this field = p		X	X		
field ID user menu default data [0]			X	X	
…			X	X	

Shared Data Element	1	2	3	4	5
field ID user menu default data [p-1]			X	X	
checksum LSB user menu data: default values	X		X	X	
checksum MSB user menu data: default values	X		X	X	
checksum LSB all Shared Data excluding user modified values	X		X		
checksum MSB all Shared Data excluding user modified values	X		X		
field ID = user menu data: modified values		X			
bytes in this field = p		X			
field ID user menu data modified value[0]				X	X
…				X	X
field ID user menu data modified value[p-1]				X	X
checksum LSB user menu data: modified values	X			X	X
checksum MSB user menu data: modified values	X			X	X
Checksum LSB of all Shared Data including user modified values	X				
Checksum MSB of all Shared Data including user modified values	X				

Table 13 - Shared Data Legend

Shared Data Legend	
1	Overhead to verify integrity of data.
2	Overhead to allow for future additions and backwards compatibility of Shared Data
3	Factory-defined – no modification allowed
4	User menu default values - identical data when programmed during Control test
5	Initially a copy of user menu default values. Modification allowed via menu mode

13.0 Annex C – Bibliography

“TIA-485 (Revision A), Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems” *Telecommunications Industry Association*, 1998.

Zimmermann, Hubert (April 1980). "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection". *IEEE Transactions on Communications*