SISO-STD-013-2014

Standard for Common Image Generator Interface (CIGI)

Version 4.0

22 August 2014

Prepared by
Common Image Generator Interface (CIGI)
Product Development Group

P.O. Box 781238
Orlando, FL 32878-1238, USA

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA

Revision History

Version	Section	Date (MM/DD/YYYY)	Description
4.0		08/22/2014	Initial published version

Participants

At the time this product was submitted to the Standards Activity Committee (SAC) for approval, the Common Image Generator Interface (CIGI) Product Development Group had the following membership and was assigned the following SAC Technical Area Director:

Product Development Group

Simon Skinner (Chair)
Ronald Moore (Vice-Chair)
Willard B. Phelps (Secretary)

Grant Bailey (SAC Technical Area Director)

Castaner, Gilbert
Chawla, Bobby
Dobrindt, Uwe
Gamble, Murray
Humphries, Roland
Irizarry, Vincent
Jones, Edward
Marrou, Lance
Murray, Bob
Peterson, Mark
Riner, Bruce
Sampson, Andrew
Savant, John
Schroeder, Curt
Siegfried, Robert
Stephens, Steve
Terry, Brett
Whitley, Chas
Wieselthier, David
Woodman, Michael

The Product Development Group would like to especially acknowledge those individuals that significantly contributed to the preparation of this product as follows:

PDG Drafting Group

Curt Schroeder (Editor)

Duff, Sean
Gamble, Murray
Humphries, Roland
Peterson, Mark
Phelps, Willard B.

The following individuals comprised the ballot group for this product.

Ballot Group

Blais, Curtis
Chawla, Bobby
Gamble, Murray
Huiskamp, Wim
Humphries, Roland
Jones, Edward
Kuiper, Paul
Marrou, Lance
Oates, William
Peterson, Mark
Phelps, Bill
Savant, John
Schroeder, Curt
Searle, Jonathan
Wong, Kwok

When the Standards Activity Committee approved this product on 15 July 2014, it had the following membership:

Standards Activity Committee

Jeff Abbott (Chair)
Marcy Stutzman (Vice Chair / Secretary)

Bailey, Grant
Blais, Curt
Blount, Elaine
Gravitz, Peggy
Gupton, Kevin
Lowe, Paul
McGlynn, Lana
McLean, Thom
Oates, William
Riggs, Bill
Youngblood, Simone

When the Executive Committee approved this product on 22 August 2014, it had the following membership:

Executive Committee

Michael O'Connor (Chair)
James Coolahan (Vice Chair)
Jane Bachman (Secretary)

Abbott, Jeff
Bouwens, Chris
Daly, John
Graham, David
Gustavson, Pau
Igarza, Jean-Louis
Morse, Katherine
Scrudder, Roy
Whittington, Eric

Introduction

Prior to this standard, image generator suppliers defined their own proprietary interface control documents for communicating with their particular products. This made compatibility of disparate image generators impossible, and caused large recurring costs to make the necessary modifications to a host simulation when migrating to another supplier's product.

The Common Image Generator Interface (CIGI) is the culmination of work begun by The Boeing Company in late 1999 in an effort to increase plug-and-play compatibility, while significantly reducing the recurring cost of integrating a wide range of image generator offerings. CIGI 1.0 was released in March 2001, followed by v. 2.0 a year later, and v. 2.1 in January 2003. The Boeing Company has provided this work as goodwill to foster these concepts.

Building on Boeing's original work, the Simulation Interoperability Standards Organization formed a CIGI Study Group in September 2003. The Study Group was well received and became a Standing Study Group, which matured the CIGI standard through SISO's established processes and the hard work and dedication of contributors throughout the visual simulation industry and user community.

TABLE OF CONTENTS

TABLE OF FIGURES

Figure Page

TABLE OF TABLES

Tables Page

1. Overview

1.1 Scope

To promote compatibility among Image Generators (IGs), this Interface Control Document (ICD) defines a common Host-to-IG interface standard. This document is not intended to define IG functional requirements. Therefore, implementation of Common Image Generator Interface (CIGI) does not imply that an IG is required to support all features addressed by the ICD, nor is it intended to limit features of an IG.

1.2 Purpose

This document describes the Common Image Generator Interface (CIGI) Version 4.0. This document contains descriptions of all data parameters, event sequences, and input/output (I/O) protocols necessary to accomplish this task using CIGI.

1.3 Objectives

Achieving compatibility among disparate systems is difficult without standards. The Modeling and Simulation (M&S) community needs a way to achieve affordable compatibility of disparate visual systems through the incorporation of a standard protocol that defines system-to-system interactions. Without a standard interface, each IG provider uses a proprietary communication schema. This results in costly reinvestment each time a visualization product is integrated into a new or existing system, unless the same visualization product is used.

This problem is exacerbated by the fact that IG hardware evolves and becomes obsolete very quickly. A standard interface to this evolving hardware will allow the replacement of obsolete hardware with more contemporary hardware solutions without incurring the cost of implementing a proprietary interface each time there is a hardware change. Furthermore, if disparate visualization products are used each interface implementation must be fashioned to a somewhat arbitrary and localized definition of common attributes, such as entity identification and methods of describing subsystem-to-subsystem interactions. It would be beneficial to all concerned if there was a standard interface for visualization tools integration. The exchange of CIGI version information between systems would further improve the potential for compatibility by allowing for automatic adjustment to a specific ICD version.

Compliance is essential to the success of any standard. To date CIGI implementers have not been constrained to comply with the ICD, i.e., an "acid test." This is unfortunate and has resulted in not all CIGI implementations being equal. This ICD seeks to define testable requirements for all functionality. The creation of automated compliance testing tools for these requirements and a suitable scoring framework is in development.

1.4 Intended Audience

This ICD has been written for software engineers involved in the integration of a host simulator with an IG and IG vendors.

1.5 Conventions Used in This Document

The following typographic conventions are used in this document:

- Data packet parameter names are *italicized*.
- Data packet names are in **boldface** type.
- Variable names are in *italicized Times Roman* typeface.
- Coordinate system axes are labeled or referenced using **boldface Times Roman** typeface.

- Hexadecimal (base-16) notation is indicated by a lower-case “h” following a numerical value (i.e.,, 8000h).
- Bit positions within each byte are numbered from right to left, indicating that the leftmost bit in each byte is the most significant bit. The leftmost byte in each word has the lowest physical address.

2. References (Normative)

2.1 SISO References

#	Document Number	Title	Date
1.	ESI	SISO-REF-010 Enumerations for Simulation Interoperability XML file, Section <cet uid="30" name="Comprehensive Entity Types">	
2.	CIGIBP	SISO-REF-046-00v1-0-d02, CIGI Implementation Techniques and Best Practices	

2.2 Other References

#	Document Number	Title	Date
3.	WEBSTER	ISBN-10: 0877798095, Merriam-Webster's Collegiate Dictionary, Eleventh Edition forward	
4.	DIS	IEEE Std 1278.1-2012 - IEEE Standard for Distributed Interactive Simulation - Application Protocols	2012
5.	ISOD2	ISO/IEC Directives, *Part 2 — Rules for the structure and drafting of International Standards.* 6th ed.	2011
6.	UTF	http://www.unicode.org/versions/latest/	

3. Definitions, Acronyms, and Abbreviations

English words are used in accordance with their definitions in the latest edition of Merriam-Webster's Collegiate Dictionary [WEBSTER] except when special SISO Product-related technical terms are required.

3.1 Definitions

Term	Definition
Animation	A predefined sequence of state changes of an entity or a portion of an entity.
Big-endian	The most significant byte of the word is stored in the smallest address containing the word and the least significant byte is stored in the largest address containing the word.
CIGI Session	An abstraction that encompasses all CIGI-related communication and corresponding state data between the Host and IG across a connection or between a pair of addressable ports.
Entity	A static or dynamic simulation object that can be created, manipulated, and destroyed by the host (see Section 5.1).
Field of View	The extent of the observable modeled world that is rendered by the IG at any given moment as defined by view angles for the top, bottom, right, and left sides and distances to front and rear clipping planes.
Frame	A real-time simulation is divided into processing frames at a fixed frequency, where all processing for the next incremental update is completed during a single frame. An IG will typically run at a frame rate to match the video vertical sync in order to reduce visible artifacts. The encompassing time period (i.e., one second) may be referred to as a major frame and the individual processing frames are minor frames.
Frustum	The region of space in the modeled world that may appear on the screen; it is the field of view (FOV) of the notional camera.
Host	The real-time simulation computer communicating with the image generator.
Host-to-IG Message	A message sent from the host to the IG containing only one IG Control packet located at the beginning.
IG-to-Host Message	A message sent from the IG to the host containing only one SOF packet located at the beginning.
Image Generator	A system that is capable of rendering a graphical scene to a display medium.
Little-endian	The least significant byte is stored in the smallest address containing the word and the most significant byte is stored in the largest address containing the word.
Message	A contiguous stream of packets sent during a single frame.
Orthographic Projection	A projection where the lines of projection are parallel to the sides of the view volume and perpendicular to the view plane.
Packet	A short fixed-length section of data that is transmitted as a unit in an electronic communications network. Specific to CIGI, a packet is a discrete data packet as defined in this ICD.
Perspective Projection	A projection onto a view plane that shows distant objects as smaller to provide additional realism as compared to an Orthographic Projection.
SEDRIS	An infrastructure technology for: (a) the representation of environmental data, and (b) the interchange of environmental data sets (ISO/IEC, 2013).
Symbol	A single drawing primitive or a group of drawing primitives that can be drawn on a symbol surface within a particular view.
Symbol Primitive	An atomic building block for a symbol, e.g., Text, Circle, Arc, Point, Line, Line Strip, Line Loop, Triangle, Triangle Strip, or Triangle Fan (and some textured variants).
Symbol Surface	A two-dimensional drawing region on a virtual plane upon which symbols may be drawn.
Symbol Template	An IG specific symbol type that can be attached to a symbol surface.

UV Mapping	This process projects a texture map onto a 3D object. The letters "U" and "V" denote the axes of the 2D texture because "X", "Y" and "Z" are already used to denote the axes of the 3D object in model space.
View	A projection of a view volume onto a 2D view plane.
View Group	A collection of views attached to a single entity.
View Plane	The 2D projection plane associated with a view and rendered to a viewport.
View Volume	The 3D volume defined by the frustum that will encompass an orthographic (box) or perspective (truncated pyramid) projection.
Viewport	A view plane rendered to a 2D region of specified size and location within a display medium.

3.2 Acronyms and Abbreviations

Acronym or Abbreviation	Meaning
2D	Two-dimensional
3D	Three-dimensional
CIGI	Common Image Generator Interface
DCS	Dynamic Coordinate System
DIS	Distributed Interactive Simulation
DOF	Degree Of Freedom
ERM	Earth Reference Model
FASCODE	Fast Atmospheric Signature Code (H.J.P. Smith)
FLIR	Forward Looking Infrared
FOV	Field of View
HAT	Height Above Terrain
HOT	Height of Terrain
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IG	Image Generator
I/O	Input/Output
IP	Internet Protocol
ISO	International Organization for Standardization
LOS	Line of Sight
LSB	Least Significant Byte
LSN	Least Significant Nibble
LSW	Least Significant Word
MODTRAN	MODerate resolution atmospheric TRANsmission (Spectral Sciences, Inc., 2012)
MSB	Most Significant Byte
MSL	Mean Sea Level
MSW	Most Significant Word
NED	North-East-Down
OTW	Out the Window
PCR	Problem/Change Request
PDG	Product Development Group
PSG	Product Support Group
SISO	Simulation Interoperability Standards Organization
SOF	Start of Frame
SRF	Spatial Reference Frame
UDP	User Datagram Protocol
UID	Unique Identification Number
UTF	Unicode Transformation Format
VSync	Vertical Sync
WGS	World Geodetic System

4. Interface Theory

The Common Image Generator Interface (CIGI) is a standardized interface between a real-time simulator host and an image generator. CIGI is an open interface offered to promote commonality in the visual simulation industry.

CIGI is a data packaging protocol and thus does not depend upon a specific physical communications medium or transport protocol. Any suitable physical medium may be used, including Ethernet, Token Ring, optical fiber, shared memory, etc. The transport protocol(s) used depend upon performance and what is appropriate to the communications hardware. This document assumes the use of the User Datagram Protocol (UDP) over Ethernet for ease of discussion.

In the most typical use case, a Host may communicate with exactly one IG as illustrated in Figure 1. The physical connection may include a switch or a hub.

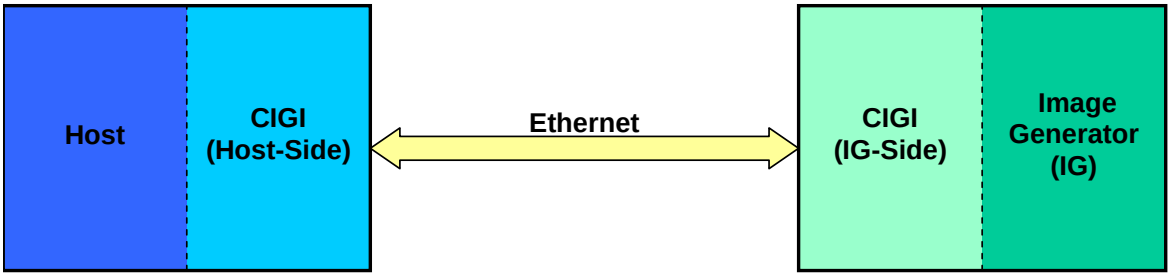


Figure 1 – Connection Using Ethernet

While it is suggested that every CIGI session (see Section 4.1) have exactly one Host and exactly one IG as shown above, some situations might justify the use of multiple receivers. The most obvious example of this type of situation would be when a Host uses UDP multicast to send data to multiple IGs to produce identical imagery on each. The Host would select one IG as a “master” and would ignore all IG-to-Host packets except those originating from the master.

It is not advisable for an IG to send data to multiple Host devices in the same session. This situation could produce undesirable results due to order dependence of some packets. Further, all mission function and response data would only be valid for one Host.

The use of multiple sessions is generally preferable to broadcasting or multicasting to multiple receivers in the same session. This way the Host may store separate state information for each IG and all IGs should communicate their operational state, mission function responses, and other data to the Host.

CIGI sessions are described in the following section.

4.1 Sessions

A CIGI *session* is an abstraction that encompasses all CIGI-related communication and corresponding state data between the Host and IG across a connection or between a pair of addressable ports. Multiple sessions’ messages may be multiplexed over the same physical network connection but shall use different addressing and/or ports to ensure that the data are demultiplexed properly.

A Host or IG may have more than one active session at one time. For example, a host trainer device might communicate to an out-the-window (OTW) IG to produce visible-light imagery. The device might also communicate with a sensor IG to produce various simulated sensor images. Further, the Host might communicate with a simulated radar generator to produce virtual radar imagery. The Host would maintain three separate sessions with these devices.

Note that the above example could be devised using a single session with multiple IGs receiving the same data over UDP multicast; however, using multiple sessions is preferable. For instance, the frequency of state data (i.e., Entity Position packets) might be different for the different IGs. Some data such as Articulated Part Control packets or Height Above Terrain (HAT)/Height Of Terrain (HOT) or Line Of Sight (LOS) requests might not need to be sent to all three IGs. Additionally, critical packets could be tracked per IG device and only re-sent to the IG indicating data loss.

The Host and IG shall each maintain unique state data for each session. For example, the **Start of Frame** and **IG Control** packets contain sequential frame number parameters (see Sections 6.2.1 and 6.1.1, respectively). At any given time, the current Host and IG frame number values in one session are completely unrelated to the current frame number values in other sessions.

4.2 Message Synchronization

CIGI supports both synchronous and asynchronous operation. Each of these modes is described below.

4.2.1 Asynchronous Operation

During asynchronous operation, the Host may send data to the IG at arbitrary intervals. The message may occur in response to a system timer or some other event within the Host. The IG may in turn send messages to the Host containing IG status and mission function data; however, the interval between Host-to-IG messages shall be determined by the Host itself.

Meanwhile, the IG should maintain a consistent frame rate. During each frame, the IG should check its buffer for incoming CIGI messages and should update its internal state based on the contents of any received messages.

Because the Host might send a message at any point during the IG's frame, positional and other state changes might not be applied until the beginning of the *next* frame. This introduces a latency of up to almost one additional frame, as shown in Figure 2.

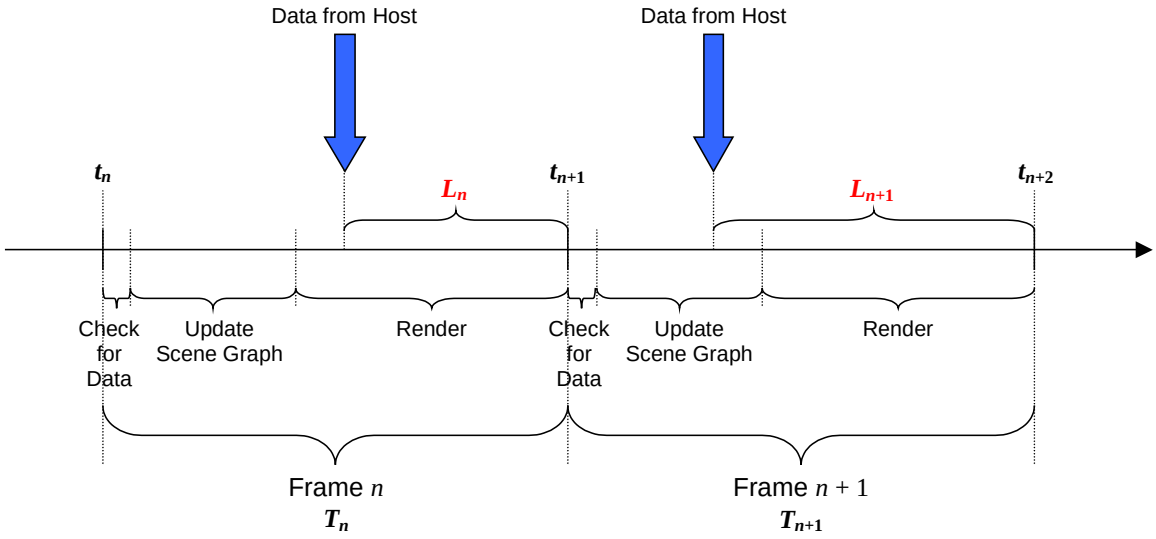


Figure 2 – Latencies Caused by Asynchronous Operation

In this example, the IG begins frame n at time t_n. At some time during that frame, the IG receives a message from the Host. Because the IG has already checked for incoming data, the message is not actually read until the start of the next frame at time t_{n+1}. By this time, the data within the message is stale by a margin of L_n and will not be represented in the scene until at least t_{n+2}, bringing the total delay to L_n + T_{n+1}. This may cause a noticeable lag in the scene.

Note that latency L_{n+1} is larger than L_n. This "creeping" of the frame offset is a common phenomenon that occurs when the Host and IG frame rates are not exactly equal. Depending upon the direction of the creep, this will frequently cause the IG to receive either zero or two Host-to-IG messages during a frame, causing frame jitter.

To reduce the effects of lag and jitter, the IG may extrapolate positional data each frame. The **IG Control** and **Start of Frame** packets (see Sections 6.1.1 and 6.2.1, respectively) include a *Timestamp* parameter that the IG may use when performing the extrapolation. This parameter indicates the amount of time that has elapsed since some initial reference time. By determining entities' velocities and accelerations from prior frames, or preferably from the **Velocity Control** and **Acceleration Control** packets (see Sections 6.1.8 and 6.1.20, respectively), the IG may calculate the probable positions of those entities during the current frame.

Because entities are often extremely dynamic, such extrapolations are prone to error. Asynchronous operation, therefore, is recommended only in low-fidelity applications or in conjunction with a shared time base [CIGIBP, Shared Time Base] when synchronous operation is not possible.

4.2.2 Synchronous Operation

During synchronous operation, the IG shall send an IG-to-Host message to the Host to indicate a new frame. This message, where the transmission is usually driven by a vertical sync (VSync) signal from the display system, functions as a "heartbeat" that dictates the timing of data transfers between the IG and Host. The IG-to-Host message also contains mission function responses, event notifications, and other IG data.

The Host shall immediately respond to each IG-to-Host message with its own message containing entity positions and orientations, component states, and other data describing changes to the scene during the previous frame. The Host shall then begin its next computational cycle, while the IG updates and renders the scene.

This mechanism makes the host data available at the beginning of every IG frame, eliminating the additional latency represented by L_n in Figure 2. Also, because the Host's frame rate will be regulated by the IG, this will mitigate jitter caused by the creeping effect of misaligned frames.

Figure 3 illustrates the sequence described above:

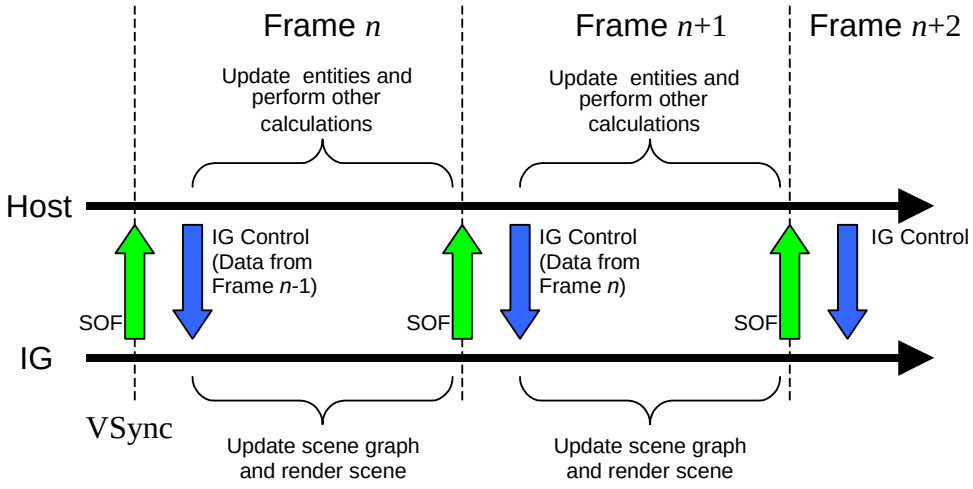


Figure 3 – Synchronous Start-of-Frame/Response Cycle

Depending upon bandwidth limitations and the transport delay, the IG might not receive a response in time to finish rendering the scene before the start of the next frame. To alleviate this situation, a time

offset may be introduced so that the IG sends each IG-to-Host message slightly *before* the beginning of the next IG frame. This technique allows data to arrive from the Host at such a time as to allow the IG its entire frame time for computations and rendering. Because the transport delay might vary from frame to frame, this offset may be adjusted to allow for worst-case network loads. Figure 4 illustrates the start-of-frame offset technique:

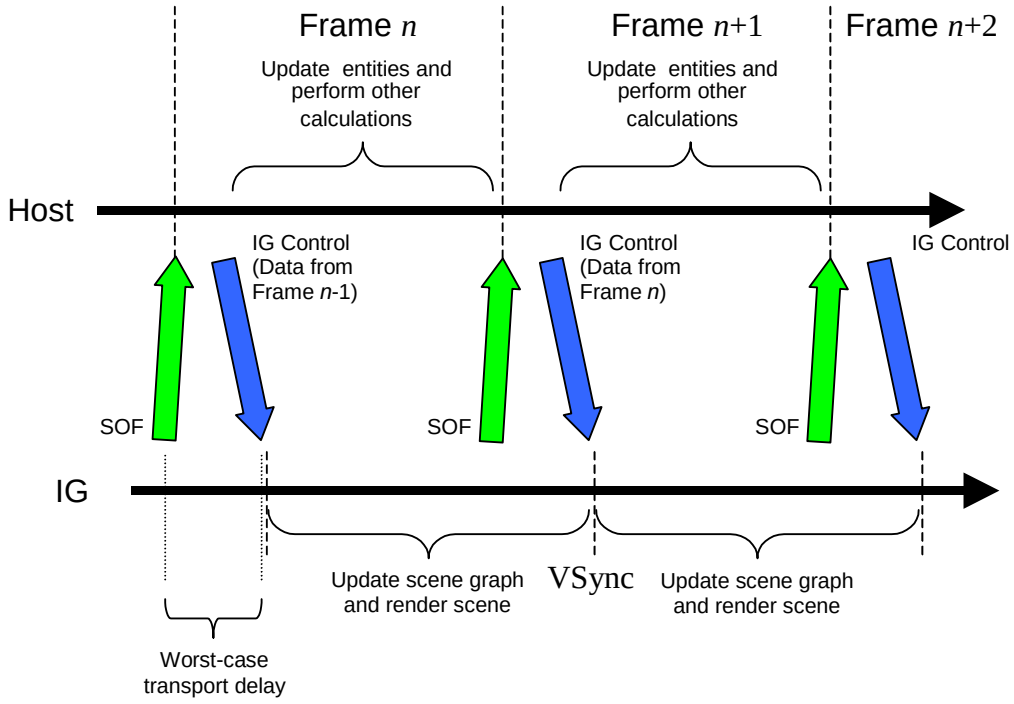


Figure 4 – Synchronous Message Timing Offset

4.3 Frame Numbering

To enable detection of lost messages in synchronous mode, both the IG-to-Host message and the Host-to-IG response message shall include sequential frame numbers. The frame number for each device is independent and should be verified by the recipient. Before sending its next message, the Host device shall increment the frame number value and populate the *Host Frame Number* parameter of the outgoing **IG Control** data packet. Upon receiving the message, the IG shall place this same value in the *Last Host Frame Number* parameter of the next outgoing **Start of Frame** packet.

Likewise, the IG device shall increment its frame number value and populate the *IG Frame Number* parameter of the outgoing **Start of Frame** data packet, and the Host shall place this same value in the *Last IG Frame Number* parameter its next outgoing **IG Control** packet.

This mechanism provides the original sender with an acknowledgement that the message in question was received. If the device detects that one or more messages were lost, it may resend critical packets or perform some other recovery action. The proper sequence of events is illustrated below:

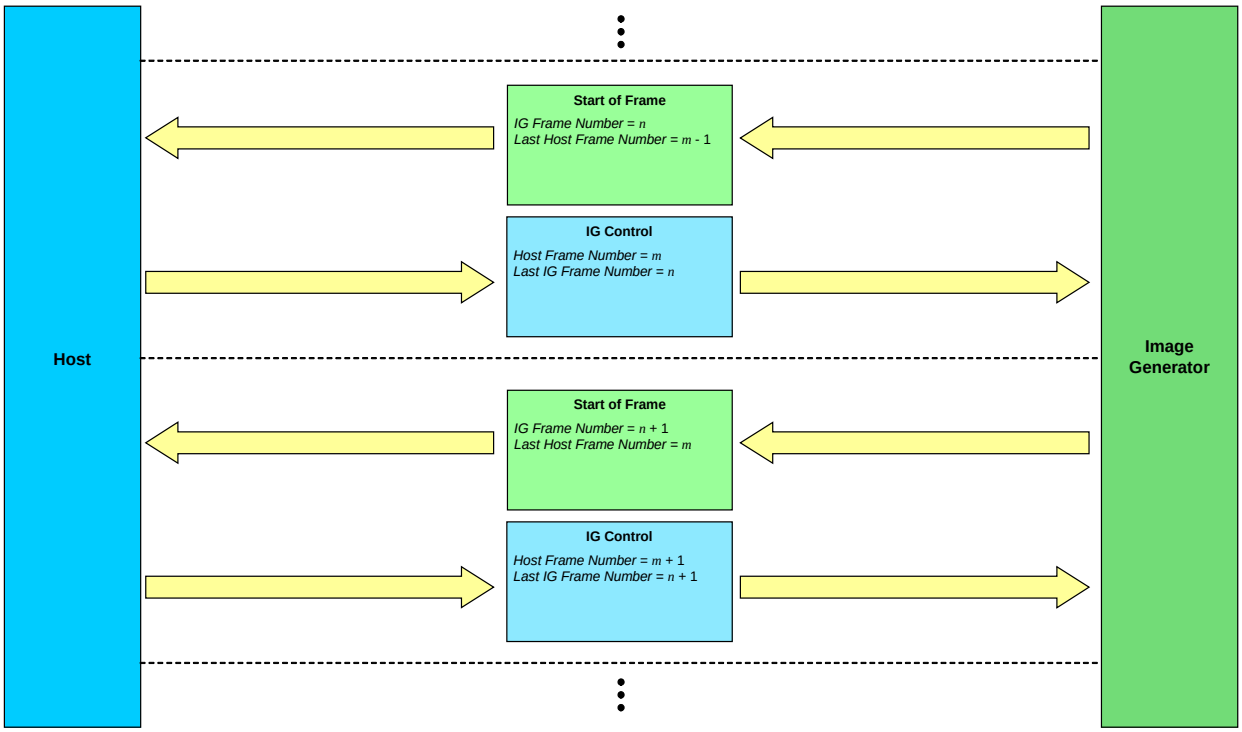


Figure 5 – Frame Numbering Sequence

This mechanism should not be used to validate message delivery of a specific message in asynchronous mode, but may be used to verify messages are being processed by the receiver in order to perform some recovery action. The *Host Frame Number* and *IG Frame Number* parameters shall still be populated as described above.

Note that in version 3.0 and 3.1 of CIGI the *Host Frame Number* and *IG Frame Number* parameters were both named “Frame Counter” and were synchronized. The value originated in the **Start of Frame** packet from the IG, and the Host copied this value to the *Frame Counter* parameter of the **IG Control** packet.

4.4 Byte Order

When a processor stores or performs an operation on a multiple-byte datum, the datum can be represented in one of two ways. On a “big-endian” platform, the lowest-addressed, or “leftmost,” byte is the high-order byte. Motorola and some other RISC architectures use this method. In “little-endian” architectures such as those used by Intel, the leftmost byte is the low-order byte.

Figure 6 illustrates the byte layout of signed and unsigned 16- and 32-bit data types on big-endian architectures. The most significant bits are highlighted.

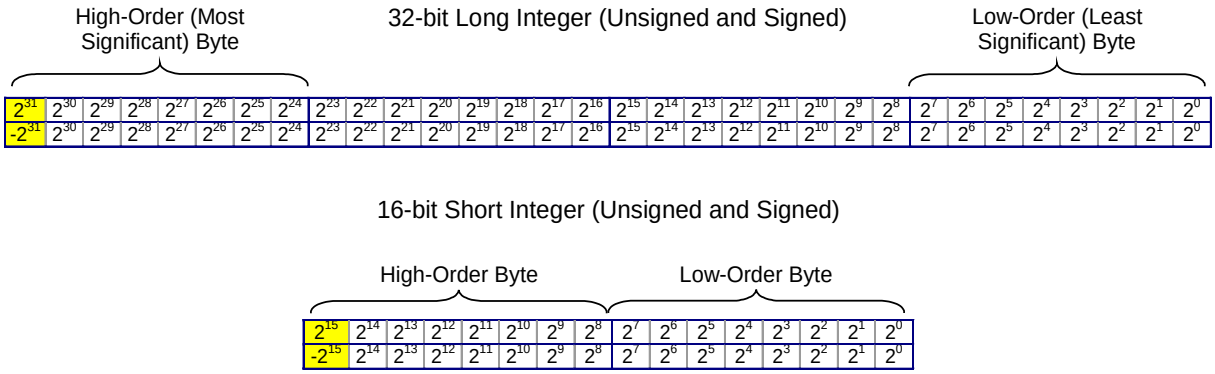


Figure 6 – Big-Endian Byte Order

Figure 7 illustrates the byte layout of the same data types on little-endian architectures:

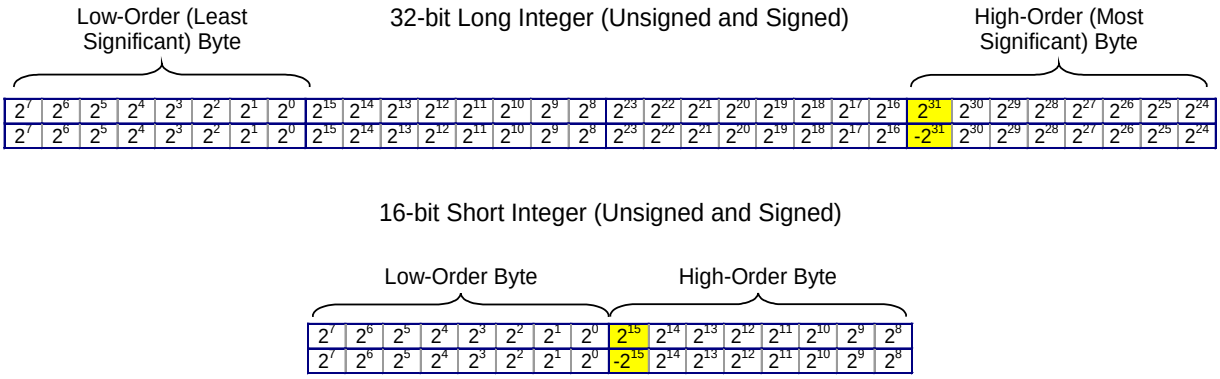


Figure 7 – Little-Endian Byte Order

CIGI 4 does not impose a byte order upon either the IG or the Host. Instead, the receiver shall perform byte swapping if necessary. This scheme has several benefits. First, no swapping is necessary if both the Host and the IG use the same native byte order. Second, it allows multiple IGs with differing byte orders to receive multicast data from a single Host (see Section 4.1). Other benefits include distribution of the additional processing between the Host and IG and simplified coding, since only a device's unpacking routines need to account for byte order.

Prior to CIGI 4, the **IG Control** and **Start of Frame** packets both contained a 16-bit *Byte Swap Magic Number* parameter that could be used to detect whether the receiver needed to byte-swap the incoming message. When using ICD versions prior to CIGI 4, the sender needed to set the most significant bit of the high byte and clear all the other bits. When the receiver examined this parameter, it would byte-swap the entire message if the most significant bit of the *low* byte was set.

CIGI 4 introduces a new packet structure that begins with a 16-bit Packet Length field. In addition to supporting longer message packets, these first two bytes also replace the *Byte Swap Magic Number* parameter. If the parser detects a zero in the "leftmost" byte, then the message is in Big Endian byte order, whereas if the zero is in the "rightmost" byte then the message is in Little Endian byte order. Figure 8 illustrates the bit locations for each state.

Big-Endian Receiver

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}			
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	⇨	18h	**No Swap**
0	0	0	1	1	0	0	0	**0**	0	0	0	0	0	0	0	⇨	1800h	**Swap**

Little-Endian Receiver

2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}			
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	⇨	1800h	**Swap**
0	0	0	1	1	0	0	0	**0**	0	0	0	0	0	0	0	⇨	18h	**No Swap**

Figure 8 – Use of CIGI Byte Swap Magic Number Parameter

The Packet Size of both the **IG Control** and **Start of Frame** packets shall be limited to 248 to ensure that a zero is contained in the high-order bytes of these packets. Successive packets in the message may exceed this size since the byte order will have already been determined. Refer to Sections 6.1.1 and 6.2.1 for additional details on the **IG Control** and **Start of Frame** packets.

4.5 Message Structure

CIGI messages are comprised of one or more data packets. During synchronous operation, there shall be exactly one IG-to-Host message and one Host-to-IG message per frame.

The first data packet in each message from the IG to the Host shall be a **Start of Frame** packet (see Section 6.2.1) and shall have a Packet ID of 0h. The message may also contain additional IG-to-Host packets as listed in Table 1.

The first data packet in each message from the Host to the IG shall be an **IG Control** packet (see Section 6.1.1) and shall have a Packet ID of FFFFh. Additional Host-to-IG packets (see Table 1) may follow within the message.

CIGI 4 introduces a four byte packet header. The first two bytes of this header contain the packet size in bytes; the next two bytes uniquely identity the packet type. The special Packet IDs assigned to the **IG Control** and **Start of Frame** packets permit a multi-version CIGI parser to differentiate between the old and new message structure as both 0h and FFh are unknown CIGI major version numbers. The remainder of each packet contains data that pertain to that particular packet. Note that all packet data that are used to uniquely identify an object on the IG or Host (i.e., a moving model or a view) are generally contained within the first eight bytes of each packet.

All 16-bit data begin on half-word (16-bit) boundaries. All 32-bit and 64-bit data begin on word (32-bit) and double-word (64-bit) boundaries, respectively.

All packets shall begin and end on 64-bit offsets within the message buffer. Packets shall be padded as necessary so that the packet size will be an even multiple of eight (8) bytes.

A CIGI implementation shall, as much as possible, package all data necessary for a particular application using standard CIGI packets while retaining the intended use as described in this ICD. If further functionality is required by the application, the intended use of the packets defined by this ICD shall be preserved but may be supplemented with extension packets.

Registered Extension Packet IDs (see Section 6.3) shall be assigned within the range 1000h - 7FFFh. Locally defined Extension Packet IDs_shall be assigned within the range 8000h – FFFEh. Standard Packet IDs shall be in the range 1h – FFFh.

Entities, regions, symbols and symbol surfaces, and other objects shall be created before they are otherwise referenced. For example, if a **Component Control** packet references an entity, that packet shall follow the **Entity Control** data packet in which the entity is instantiated. If both packets occur in the same message, the **Entity Control** packet shall precede the **Component Control** packet.

Similarly, objects that have been deleted should no longer be referenced unless new objects are created with the same identifiers. For instance, if a message contains an **Entity Control** packet with the effect that Entity n is destroyed, then subsequent packets or messages should not reference Entity n until a new Entity n is created. When an entity is destroyed, it shall not be re-created until a subsequent Host-to-IG message. Packets referencing non-existent objects shall be ignored unless the packet may be used to instantiate the object.

Other than the requirements described above, no restrictions are imposed on packet ordering.

To reduce the risk of overloading the IG computational frame, an attempt should be made to minimize the amount of data contained in each message. Therefore, unless a packet is mandatory (see Table 1), only those packets containing new data should be contained within each message. For example, if an entity's position, orientation, or other attributes have not changed since the previous Host-to-IG message, the Host should not send an **Entity Position** packet.

Table 1 – Data Packet Summary

Packet ID	Data Packet Name	Mandatory Each Frame	Section
	HOST TO IG		
0h	**IG Control**	Yes	6.1.1
01h	**Entity Position**	No	6.1.2
02h	**Conformal Clamped Entity Position**	No	6.1.3
03h	**Component Control**	No	6.1.4
04h	**Short Component Control**	No	6.1.5
05h	**Articulated Part Control**	No	6.1.6
06h	**Short Articulated Part Control**	No	6.1.7
07h	**Velocity Control**	No	6.1.8
08h	**Celestial Sphere Control**	No	6.1.9
09h	**Atmosphere Control**	No	6.1.10
0Ah	**Environmental Region Control**	No	6.1.11
0Bh	**Weather Control**	No	6.1.12
0Ch	**Maritime Surface Conditions Control**	No	6.1.13
0Dh	**Wave Control**	No	6.1.14
0Eh	**Terrestrial Surface Conditions Control**	No	6.1.15
0Fh	**View Control**	No	6.1.16
010h	**Sensor Control**	No	6.1.17
011h	**Motion Tracker Control**	No	6.1.18
012h	**Earth Reference Model Definition**	No	6.1.19
013h	**Acceleration Control**	No	6.1.20
014h	**View Definition**	No	6.1.21
015h	**Collision Detection Segment Definition**	No	6.1.22
016h	**Collision Detection Volume Definition**	No	6.1.23
017h	**HAT/HOT Request**	No	6.1.24
018h	**Line of Sight Segment Request**	No	6.1.25
019h	**Line of Sight Vector Request**	No	6.1.26
01Ah	**Position Request**	No	6.1.27
01Bh	**Environmental Conditions Request**	No	6.1.28
01Ch	**Symbol Surface Definition**	No	6.1.29
01Dh	**Symbol Text Definition**	No	6.1.30
01Eh	**Symbol Circle Definition**	No	6.1.31
01Fh	**Symbol Polygon Definition**	No	6.1.32
020h	**Symbol Clone**	No	6.1.33
021h	**Symbol Control**	No	6.1.34
022h	**Short Symbol Control**	No	6.1.35
023h	**Symbol Textured Circle Definition**	No	6.1.36
024h	**Symbol Textured Polygon Definition**	No	6.1.37
025h	**Entity Control**	No	6.1.38
026h	**Animation Control**	No	6.1.39

Packet ID	Data Packet Name	Mandatory Each Frame	Section
	IG TO HOST		
FFFFh	**Start of Frame**	Yes	6.2.1
0FFFh	**HAT/HOT Response**	No	6.2.2
0FFEh	**HAT/HOT Extended Response**	No	6.2.3
0FFDh	**Line of Sight Response**	No	6.2.4
0FFCh	**Line of Sight Extended Response**	No	6.2.5
0FFBh	**Sensor Response**	See Packet Description	6.2.6
0FFAh	**Sensor Extended Response**	See Packet Description	6.2.7
0FF9h	**Position Response**	No	6.2.8
0FF8h	**Weather Conditions Response**	No	6.2.9
0FF7h	**Aerosol Concentration Response**	No	6.2.10
0FF6h	**Maritime Surface Conditions Response**	No	6.2.11
0FF5h	**Terrestrial Surface Conditions Response**	No	6.2.12
0FF4h	**Collision Detection Segment Notification**	No	6.2.13
0FF3h	**Collision Detection Volume Notification**	No	6.2.14
0FF2h	**Animation Stop Notification**	No	6.2.15
0FF1h	**Event Notification**	No	6.2.16
0FF0h	**Image Generator Message**	No	6.2.17
	EXTENSION DATA PACKETS		
1000h – 7FFFh	**Registered Extension Packets**	Application-Dependent	6.3
8000h – FFFEh	**Locally Defined Extension Packets**	Application-Dependent	6.3

Only a subset of these data packets may be required in any given message to describe data changes to the IG. Figure 9 shows a hypothetical sequence of Host-to-IG messages, each beginning with a **Start of Frame** or **IG Control** packet. Note that Host-to-IG messages are depicted as flowing left to right, so packets within the message are arranged right to left. In other words, the rightmost packet occurs first in each message.

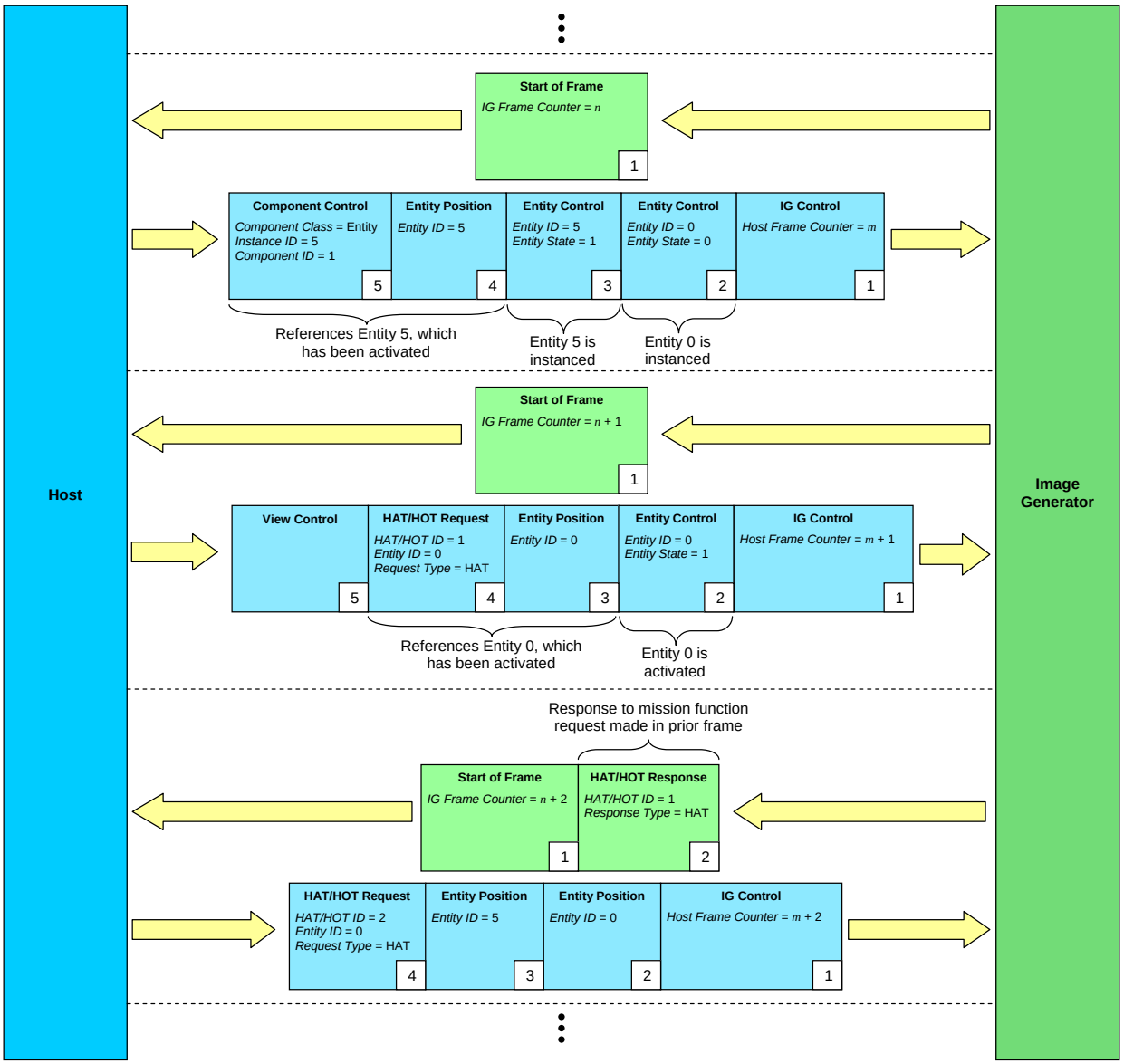


Figure 9 – Example of Message Exchange in Synchronous Mode

In the above example, the top Host-to-IG message contains **Entity Control** packets that result in Entity 0 and Entity 5 being created. Entity 5 is activated at the time of creation and can be referenced in a subsequent packet, even if that packet is contained within the same message. Entity 0 is not activated until the middle Host-to-IG message, after which it can be referenced in a subsequent packet within the same message (see Section 6.1.38).

4.6 Data Types

CIGI is platform- and language-independent. Therefore, this document uses a generic nomenclature for data types. Although some languages may use the same data type (and keyword) to represent more than one kind of datum, this document distinguishes between semantic uses.

4.6.1 Basic Types

Real values shall be represented as signed single- or double-precision floating-point numbers as defined in ANSI/IEEE STD 754-1985. This document uses the names **single float** and **double float** to refer to single- and double-precision numbers, respectively.

Integer values shall be represented as signed or unsigned 8-bit, 16-bit, or 32-bit fields. This document refers to these data types as **int8**, **int16**, and **int32**, respectively. Integers may also be represented by unsigned *n*-bit fields as described in Section 4.6.3.

Alphanumeric characters shall be represented using the Unicode Transformation Format 8-bit (UTF-8) encoding scheme as described in Section 4.6.2.

Fields with no explicitly defined type, such as the user data fields in the **Component Control Packet**, are indicated as 32-bit **word** fields. Word fields shall be treated by the interface as unsigned int32 data. If byte swapping is necessary, word fields shall be byte-swapped as 32-bit integers.

Table 2 – Basic Data Type Summary

Data Type Name	Bytes	Precision	Minimum Value	Maximum Value
octet	1	N/A	0	255
int8	1	N/A	-128	127
unsigned int8	1	N/A	0	255
int16	2	N/A	-32,768	32,767
unsigned int16	2	N/A	0	65,535
int32	4	N/A	-2,147,483,648	2,147,483,647
unsigned int32	4	N/A	0	4,294,967,295
word	4	N/A	0	4,294,967,295
single float	4	7 digits	$1.4012980 \times 10^{-45*}$ $-3.4028235 \times 10^{38\dagger}$	$3.4028235 \times 10^{38*}$ $-1.4012980 \times 10^{-45\dagger}$
double float	8	15 digits	$4.940656458412465 \times 10^{-324*}$ $-1.797693134862315 \times 10^{308\dagger}$	$1.797693134862315 \times 10^{308*}$ $-4.940656458412465 \times 10^{-324\dagger}$

* indicates range for positive values
† indicates range for negative values

4.6.2 UTF-8 Strings

Alphanumeric characters shall be represented as Unicode code points using the UTF-8 encoding scheme. UTF-8 is backward compatible with the ASCII character set (ANSI_X3.4-1986). Full details of the UTF-8 standard can be found at [UTF].

CIGI implementers need to be careful to avoid creating malformed UTF-8 strings. This is most likely to be an issue when truncating a string to fit within a single packet. Because UTF-8 code points might span more than one byte, the CIGI implementation shall ensure that such code points are not split when truncating a UTF-8 string.

Because of the broad scope of the UTF-8 standard, implementers may choose to implement only part of the full character set. Regardless of the completeness of the UTF-8 implementation, the Host or IG shall be fault-tolerant of any and all UTF-8 strings, even if some or all of the code points are ignored.

4.6.3 *n*-Bit Fields

To reduce packet size and Ethernet traffic, Booleans, enumerated values, and integers with small ranges typically use only the bits they need. Memory may be divided into small, sub-byte fields to allow the data to be "packed" into as small a space as possible. The following byte diagram shows an example of how several values might be packed into such fields:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
D3				*D2*			*D1*	Reserved					*D5*		*D4*	*D6*															

Figure 10 – Example of Multiple *n*-Bit Fields

The above word can be broken down as follows:

- *D1* is a single-bit binary value with possible values 0 and 1.
- *D2* is a 3-bit unsigned integer value with a range of 0 through $2^3 - 1$.
- *D3* is a 4-bit unsigned integer value with a range of 0 through $2^4 - 1$.
- *D4* is a single-bit binary value with possible values 0 and 1.
- *D5* is a 2-bit unsigned integer value with a range of 0 through $2^2 - 1$.
- *D6* is a 16-bit integer (signed or unsigned).
- The bits marked “Reserved” are unused and shall be set to zero by the sender.
- *n*-bit fields shall not cross byte boundaries.

4.7 Startup and Shutdown Sequences

CIGI supports four IG modes: Reset/Standby, Operate, Debug, and Offline Maintenance.

While the IG initializes, it shall disregard any CIGI messages from the Host. During synchronous operation, the Host shall communicate with the IG only in response to a IG-to-Host message. Because this restriction is not enforced in asynchronous mode, the IG shall disregard any data received from the Host during this time.

During initialization, the IG may load a pre-configured default database or test pattern. It may also pre-load entity models so that they may be instanced quickly. The IG should allow a sufficient amount of time to allow the display system and other components to come online. This time may be configurable.

When the IG completes its initialization sequence, it shall go into the “standby” state and begin sending IG-to-Host messages to the Host. The *IG Mode* parameter within each **Start of Frame** packet (see Section 6.2.1) shall be set to Reset/Standby to indicate this operational mode. By sending its first IG-to-Host message, the IG shall declare itself to be mission-ready.

The IG shall remain in Reset/Standby mode until it receives an **IG Control** packet (see Section 6.1.1) from the Host in which the *IG Mode* parameter is set to Operate. The Host shall then wait for the IG to set a **Start of Frame** packet’s *IG Mode* parameter to Operate before sending packets of any other type. The Host shall continue to send **IG Control** packets while waiting for the IG to transition from Reset/Standby to Operate. During this transition the IG shall ignore all data except the **IG Control** packets’ *IG Mode* and *Last Host Frame Number*.

When the IG sends a **Start of Frame** packet in which the *IG Mode* parameter is set to Operate, the Host may begin sending initialization and/or mission data. Initialization data may include requests to load a

new database, modifications to the views, system component controls, etc. Mission data may include entity states, non-system component controls, mission function requests, etc.

After the operational training or gaming session terminates, the Host should command the IG back to the Reset/Standby mode. The IG shall then revert to its initial mission-ready condition; all entity instances shall be removed from the scene; all in-process mission function requests shall be purged; and all views, view groups, non-entity components, and environmental and weather conditions shall be reset to their default states. Any cached databases or cached entity models may remain in memory. The simulation date and time of day may remain unchanged.

The Offline Maintenance mode shall not be initiated through CIGI. This mode is only supported in CIGI to the extent that the *IG Mode* parameter of the **Start of Frame** packet may indicate this as being the current IG state.

4.8 Control Abstraction

Because CIGI is by design a generic interface, it may not be possible to map all conceivable functional controls to a unique data packet. The **Component Control** packet (see Section 6.1.4) is provided as a general-purpose packet that may be used to control a variety of model, terrain, system, and other components. Given the necessary control function definition documentation, the integration engineer should be able to map virtually any Host or IG function to a component control. Such documentation should contain identifier and parameter assignments for each function.

4.9 Interface Extensions

Although CIGI is a robust interface, there may be times when an implementer needs to define a unique data packet format for a specific purpose. For this reason, CIGI has been designed to be extensible. Data packet IDs have been reserved for extension packets.

As noted in Section 4.5, implementers shall not duplicate existing data or functionality in extension packets. Further, the implementation shall not rely upon locally defined extension packets for basic operation.

A Host or IG device shall gracefully ignore any extension packets it does not recognize.

Refer to Section 6.3 of this document for details on registering and using extension packets.

4.10 Cross-Version Compatibility

All CIGI messages contain a major version number identifying the version of CIGI to which the packets in that message conform. This version number is contained in the *Major Version* parameter of both the **IG Control** (Section 6.1.1) and **Start of Frame** (Section 6.2.1) packets. When a device receives a CIGI message, it should inspect the major version number to ensure that both devices are compatible.

A minor version number is also contained in the *Minor Version* parameter of both the **IG Control** and **Start of Frame** packets. An increment of this minor version number indicates that changes may have been made to one or more data packets. Data packets should be able to be parsed by implementations of previous minor versions within the same major version. A device whose counterpart uses an earlier minor version may look at the *Minor Version* parameter to determine whether some features and behaviors are not supported.

A major design goal introduced in CIGI 3 was to build in provisions for limited cross-version, packet-level compatibility in anticipation of future generations of the interface. This document establishes some guidelines in order to attempt to allow a certain degree of forward- and backward-compatibility between minor versions starting at Version 3 and continuing in Version 4:

1. Sub-byte ("*n*-bit") fields shall begin at the least significant bit available within the byte.

2. All bits marked "Reserved" shall be set to zero (0) by the sender.

3. If a device receives a packet whose packet ID is not recognized, the packet shall be gracefully ignored.

4. Any parameters added to a packet in future minor versions of CIGI should utilize existing Reserved bits, if appropriate, or be appended to the end of the packet. If the receiver encounters a packet whose *Packet Size* parameter is larger than expected, any bytes beyond the expected size shall be ignored.

5. If a parameter's size needs to be increased, it should utilize any adjacent Reserved space or be moved to the end of the packet as a new parameter.

6. If a packet becomes obsolete, that packet's identifier shall not be reused within the major version.

These guidelines are not intended to impose limitations that would be detrimental to the maturation of the interface. Although every attempt should be made to follow these guidelines when making future changes to CIGI, these are not strict rules and should not take precedence over efficient design.

Note that changes made to versions of CIGI prior to Version 3.0 do not adhere to the above guidelines.

4.11 Obsolete Packets

Packets may be deprecated if, by a general consensus, the packet is determined to be obsolete or to hold little value to the CIGI community. The PDG may assign deprecated status according to the Problem/Change Request process of the PDG.

The use of deprecated packets by new CIGI implementations is discouraged; however, deprecated packets should not be removed from future revisions of the ICD with the same major version number.

Neither the Host nor IG shall be required to implement packets marked as "Deprecated." However, both the Host and the IG shall gracefully ignore any deprecated packets that are not implemented per Guideline 3 in the previous section.

5. Fundamental Concepts

5.1 Entities

An *entity* as defined in this document exhibits all of the following characteristics:

1. The entity has a local coordinate system as defined in Section 5.4.2.
2. The entity has a separate and unique model hierarchy within the IG. This hierarchy may or may not include geometry.
3. The entity's model hierarchy may be transformed (i.e.,, translated, rotated, and scaled) independently from other model hierarchies within the IG.
4. The entity's child model hierarchy may be moved anywhere within the IG's entity hierarchy. In other words, it may be attached to (become a branch of) and later detached from another entity's model hierarchy.

Entities are able to represent both static and dynamic objects. Some examples of dynamic entities are vehicles such as aircraft, ships, and automobiles; animations such as explosions, missile trails, and smoke; and localized weather phenomena such as clouds and fronts. Static objects may include stationary ground targets such as buildings and bridges, and other objects such as video test patterns.

Although terrain, weather layers, views, and articulated parts share many characteristics with entities, they are not treated as such by CIGI.

Terrain has its own hierarchical representation and coordinate system, but CIGI does not expressly allow terrain to be transformed or attached to other entities. Entities may, however, be used to represent certain parts of the terrain such as dynamic sea states and surface conditions. Although the preferred method of implementing these features is through the **Maritime Surface Conditions Control** and **Terrestrial Surface Conditions Control** packets (Sections 6.1.13 and 6.1.15, respectively), many legacy systems use terrain databases containing holes that are filled by smaller dynamic models.

Even though clouds and other discrete weather phenomena may be implemented as entities, weather layers may not. An IG might generate polygonal surfaces to represent the top and bottom of a weather layer, but these surfaces do not have distinct hierarchical models or coordinate systems.

A view does possess a distinct coordinate system, but it is derived in part from the entity it is attached to. View groups are hierarchical collections of views.

An articulated part cannot be detached from its respective model hierarchy and consequently is not an entity. Moving parts on a model may be implemented as separate models rather than articulated sub-models. In these instances, each moving part would be a child entity of the associated parent model.

Once an entity has been instantiated by an **Entity Control** packet the IG shall allow the associated *Entity ID* to be referenced within the same and subsequent frames until an **Entity Control** packet destroys the entity. See Figure 9 and Figure 11 for example usage and Sections 6.1.38 and 6.1.2 for detailed information on how the Host instantiates and controls entities on the IG with the **Entity Control** and **Entity Position** packets.

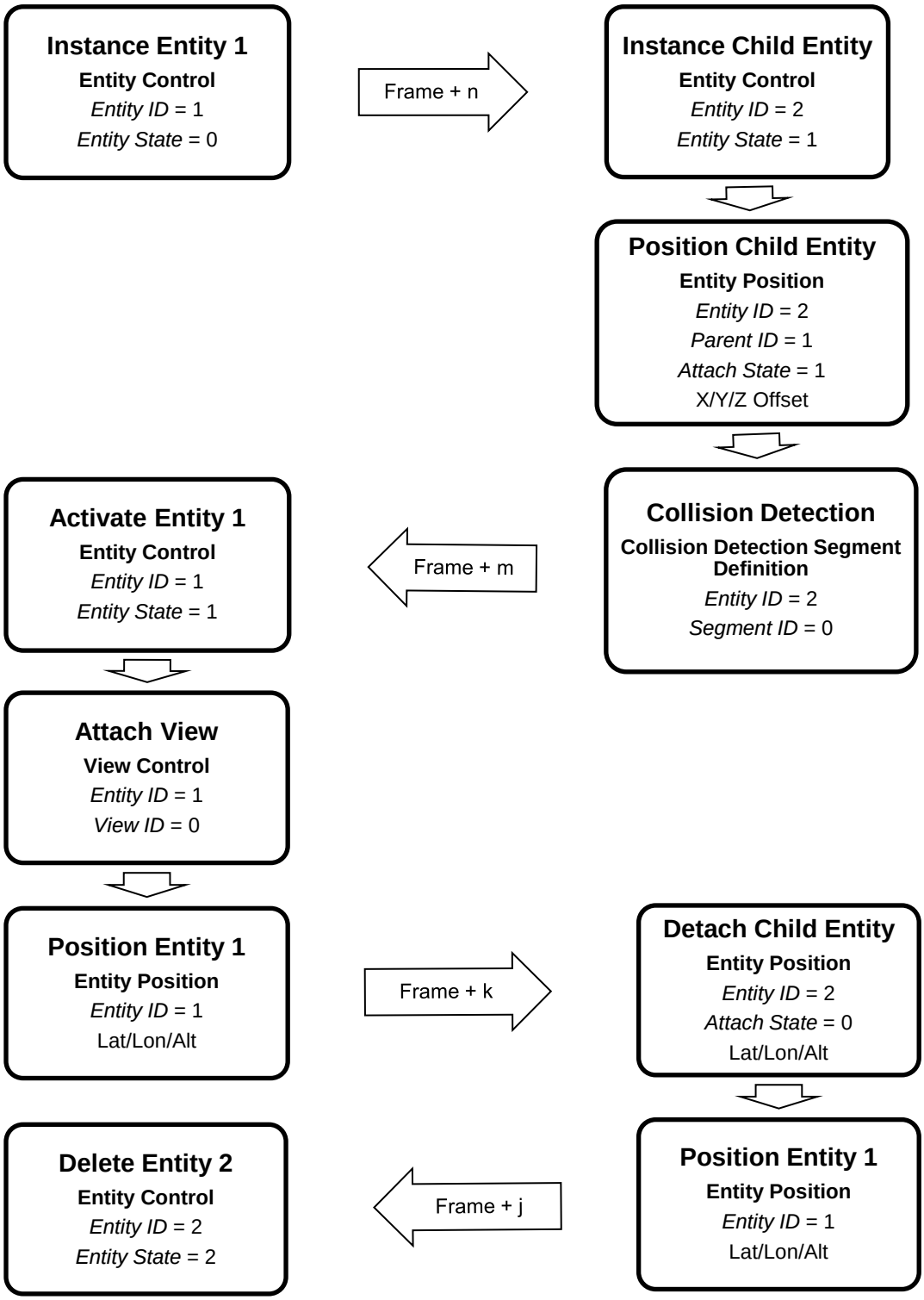


Figure 11 - Entity Creation, Control, and Deletion

Figure 11 shows a subset of the logic flow one might use configuring the entities shown in Figure 112. The airplane entity (*Entity ID* = 1) is initially instantiated as inactive (*Entity State* = 0).

Frame + n: a missile entity (*Entity ID* = 2) is attached to the airplane as a child entity (*Attach State* = 1) with an X, Y, and Z offset from the parent entity. The child entity is instantiated as active (*Entity State* = 1), but is not visible due to the parent being inactive. Similar commands would be used to attach the missile trail child entity (*Entity ID* = 4) to the missile entity in an inactive state (*Entity State* = 0). A collision segment is also defined for the missile entity.

Frame + m: the airplane entity is activated (*Entity State* = 1), thereby making the airplane and missile entities visible in the scene, but not the missile trail entity. A view (*View ID* = 0) is attached to the airplane entity and an initial position is sent for the airplane entity.

Frame + k: the missile is launched, so the host detaches the missile entity from the airplane entity (*Attach State* = 0) and an initial position in world coordinates is sent for the missile entity in addition to updating the airplane position (i.e., the missile entity does not inherit the parent entity's position as its initial position). The missile trail child entity is also activated (*Entity State* = 1) by the host to make it visible in the scene.

Frame + j: a collision is detected for the missile entity and the host removes the missile instance and its child missile trail instance from the scene by destroying the missile entity (*Entity State* = 2).

5.1.1 Animations

An *animation* is an object that has a specific pattern of movement, growth, or other behavior. This behavior is typically representative of some real-life phenomenon that, once initiated, cannot be directly controlled. Examples include fire, explosions, smoke, moving water, etc. Aside from an element of randomness introduced by stochastic processing within the IG, animations are typically reproducible for a given environmental state.

An animation object is attached to an *Entity* using an **Animation Control** packet (see Section 6.1.39). Note that some animated components may be implemented as sub-models of an entity model rather than as separate animation objects. Such components might include flashing lights, aircraft propellers, afterburners, landing gear, tank tracks, wheels, and like items. These components would be controlled with the **Component Control** packet (Section 6.1.4).

Depending upon the nature of the animation, the Host may modify certain characteristics in the **Animation Control** packet or by sending one or more **Component Control** packets (Section 6.1.4). The Host might define the expansion rate of an explosion, for example, or the length of a contrail. The specific behavior of an animation, however, is automated by the IG.

An IG may implement any of a variety of animation types. For example, a flip-book animation is a 2-D animation applied to some surface. Each texture may be one of a fixed sequence of images or may be generated procedurally each frame. The IG advances the image at some frame rate. The same concept may be applied to bump map and displacement map animations.

A frame-based geometry animation is an entity with multiple sets of geometry that are displayed sequentially. Each set of geometry is typically attached to a switch node. To display the next set, or "frame," the current switch node is switched off as the next one is switched on. The animation may consist of few frames with simple geometry (i.e., a turning propeller might be implemented as two alternating polygons with temporally aliased textures) or many frames with complex geometry (e.g., a collapsing bridge might have damaged, undamaged, and many intermediate states).

A transformation-based geometry animation, or shape animation, is an entity whose geometry is changed over time. A high-fidelity hierarchical model, for example, might contain sections of geometry that are

translated and rotated to simulate movement from wind. A cloud of black smoke produced by flak might expand and drift as its dynamic texture or particle system appears to fade. The latter example illustrates a combination of animation techniques.

Other types of animation might include, but are not limited to, motion-path animations, particle systems, and palette animations.

5.2 Views

A *view* is a projection of a three-dimensional volume onto a two-dimensional viewing plane. CIGI 4 supports perspective and orthographic parallel views. Oblique parallel views are not explicitly supported but may be implemented by using the **Component Control** packet (Section 6.1.4) to specify the direction of projection relative to the projection plane. At least one active entity shall exist that has a Viewing Volume or View Group attached in order to visualize a scene on the IG.

5.2.1 Viewing Volumes

CIGI supports perspective and orthographic parallel viewing volumes. These are described below.

5.2.1.1 Perspective

The viewing volume for a *perspective* projection is the frustum defined by the near and far clipping planes and the sides of the viewport. The viewport is an area on the view plane onto which objects within the view frustum are projected. The viewport generally corresponds to the display or a window. CIGI therefore assumes its size to be fixed. Figure 12 illustrates the concept of perspective projection of a three-dimensional viewing volume onto a two-dimensional viewport.

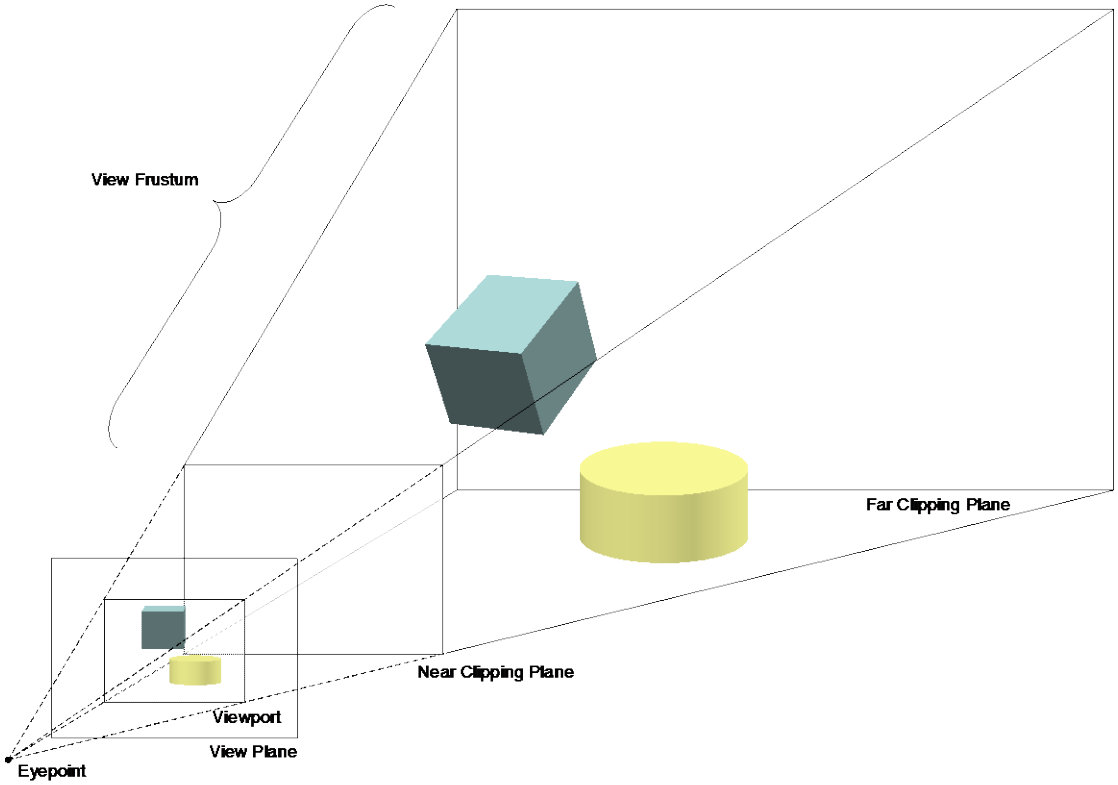


Figure 12 – Perspective Projection onto a Viewport

Because the size of the display (and thus the viewport) is fixed, the dimensions of the view frustum are directly related to the position of the eyepoint. By designing the view eyepoint to correspond to the observer's eye, a projection with an apparent one-to-one scale is achievable.

Figure 13 illustrates how a view is defined. Angle α is the left half-angle and angle β is the right half-angle. Angles γ and δ represent the top and bottom half-angles, respectively. The viewing vector corresponds to the observer's line of sight and is perpendicular to the view plane.

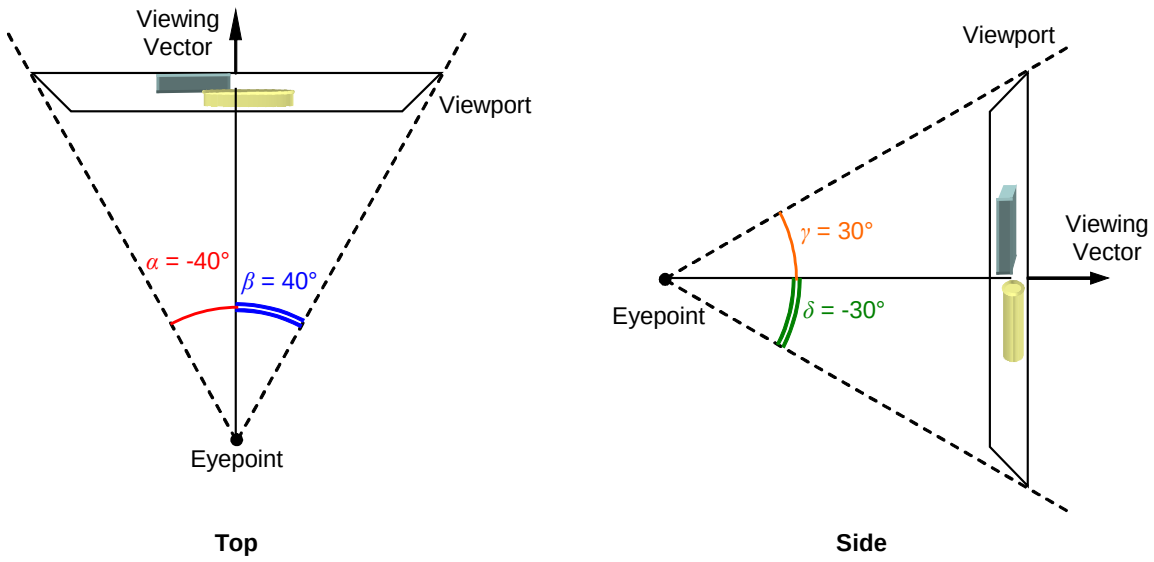


Figure 13 – View Definition Half-Angles

A view may be defined so that the sizes of angles α and β, and of angles γ and δ, are not equal. This produces an oblique view such as the one shown below:

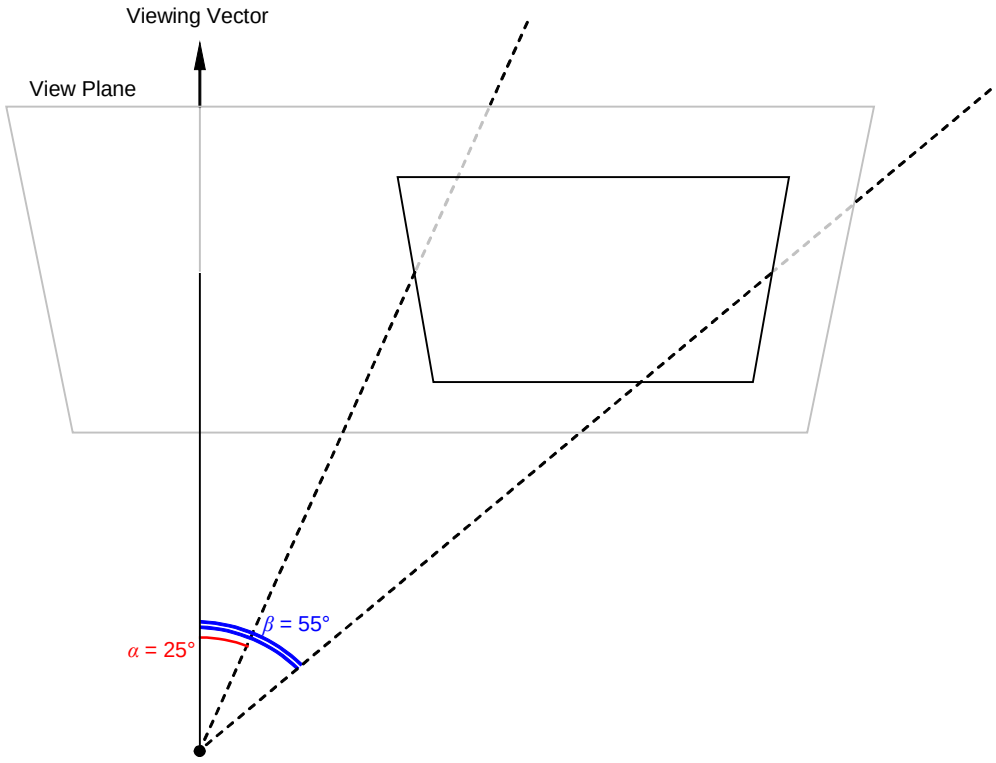


Figure 14 – Example of an Oblique Perspective View

The size of the view frustum may be set via the **View Definition** packet. Refer to Section 6.1.21 for details about this packet.

5.2.1.2 Orthographic Parallel

An *orthographic parallel* projection is one where the lines of projection are parallel to the sides of the viewing volume and perpendicular to the projection plane. This type of view is typically used for two-dimensional views such as “God’s eye” views and heads-down displays. Figure 15 illustrates an orthographic projection of a three-dimensional viewing volume onto a viewport.

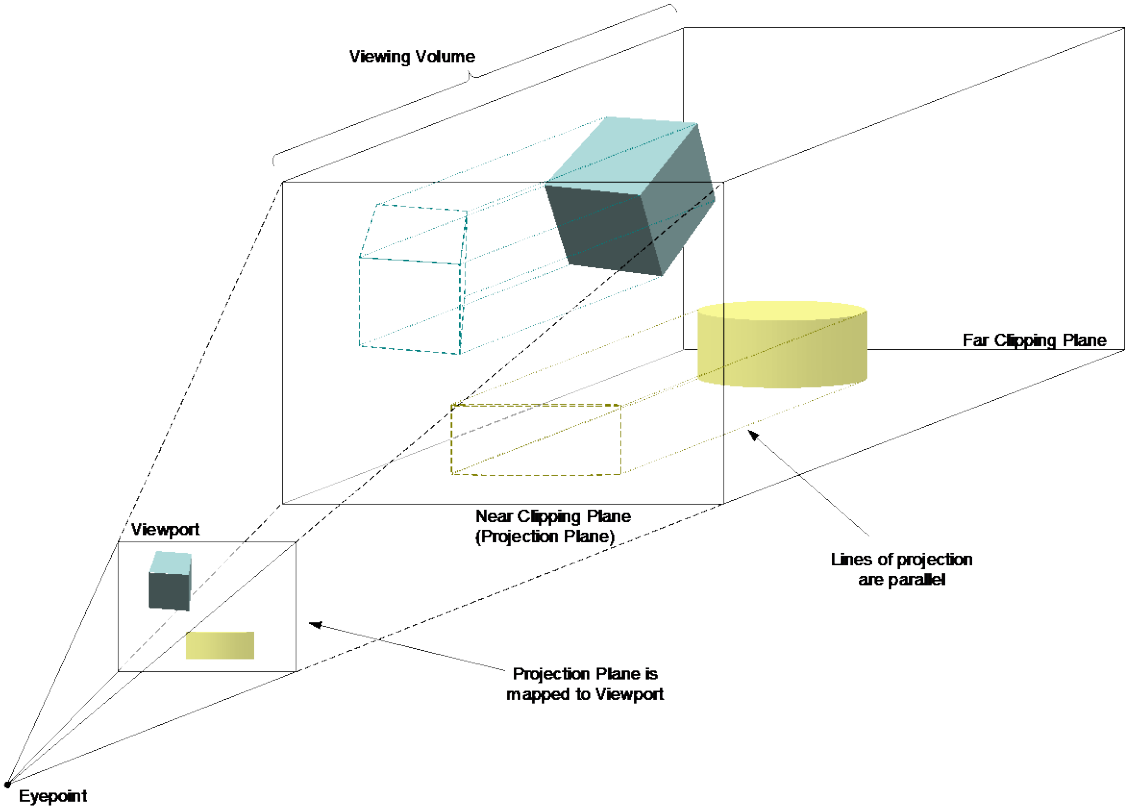


Figure 15 – Orthographic Parallel Projection onto a Viewport

The scene within the viewing volume is projected onto the projection plane. Because the lines of projection are parallel, the apparent sizes of any objects within the viewing volume do not vary with distance from the projection plane.

The projection plane is mapped to the viewport, where the projected scene is displayed.

As with a perspective projection, the width and height of the viewing volume of an orthographic projection is directly proportional to the sizes of the angles formed at the eyepoint between the viewport sides and the viewing vector (see Figure 13). The depth of the volume is the distance between the near and far clipping planes.

5.2.2 View Groups

Figure 16 below shows three views forming a panoramic scene. Certain situations may require the views to be moved spatially or hierarchically with respect to the entity. These changes may be applied to each of the three views individually. Alternatively, the views may be grouped so that operations may be performed upon them simultaneously.

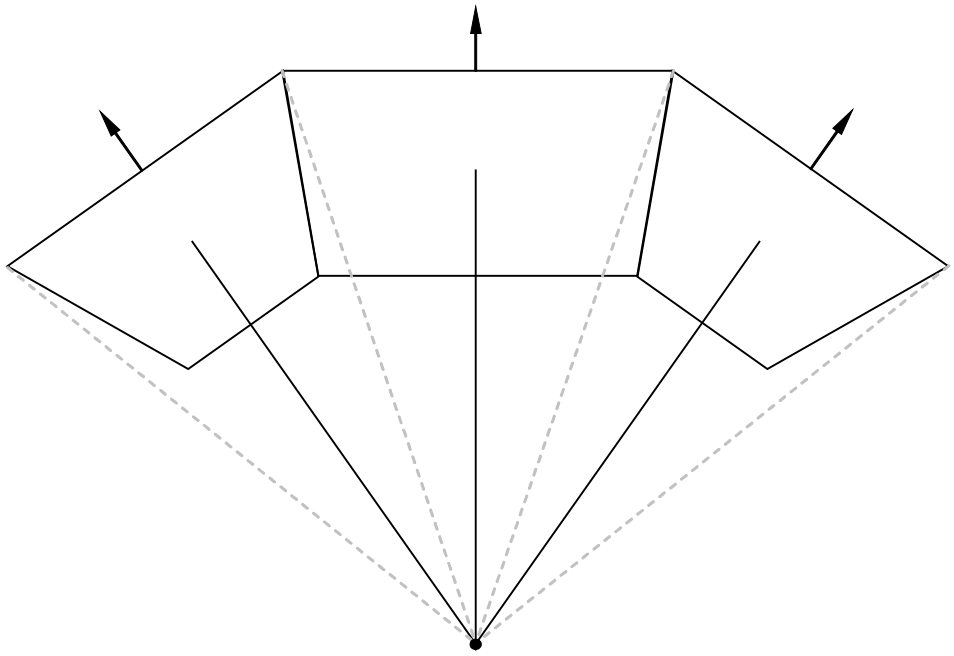


Figure 16 – Example of a View Group

The latter technique has several advantages. Perhaps most importantly, it reduces the network load because a single **View Control** packet (see Section 6.1.16) may be used instead of several. Another advantage is that it guarantees that the spatial relationships between views are maintained. In addition, the Host does not have to keep track of multiple coordinate systems for the views.

A view is assigned to a view group by setting the *Group ID* parameter of the **View Definition** packet (see Section 6.1.21). Once the group assignment has been made, the view may be controlled with the **View Control** packet (see Section 6.1.16) either individually or as part of the group. Refer to the indicated sections for more information on these packets.

5.3 Symbols

A *symbol* is a single drawing primitive or a group of drawing primitives that may be drawn on a symbol surface within a particular view.

Symbols may be defined by the Host using one of the symbol definition packets as described in Sections 6.1.30 through 6.1.32, 6.1.36, and 6.1.37. Each symbol is identified by a unique symbol ID. Each instance of a symbol shall be defined independently. If the Host uses two visually identical symbols at the same time, both symbols shall be defined and given a unique *Symbol ID* value. The Host may define a "blank" symbol for use as a top-level parent (i.e.,, root node) for grouping.

5.3.1 Symbol Surfaces

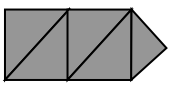
A *symbol surface*, or simply *surface*, is a rectangular, two-dimensional drawing region on a virtual plane on which symbols may be drawn. A surface is placed in 3D space relative to a particular entity or coincident with the near clipping plane of a particular view. The extents of the rectangular drawing region are defined with respect to one of the coordinate systems described in Sections 5.4.4.1 through 5.4.4.3.

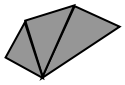
5.3.2 Symbol Primitives

Primitives are the atomic building blocks of a symbol. A single primitive may comprise a simple symbol, or multiple primitives may be combined to form more complex shapes.

CIGI defines symbol primitives to include Text, Circle, Arc, Point, Line, Line Strip, Line Loop, Triangle, Triangle Strip, or Triangle Fan. These are described in Table 3.

Table 3 – Symbol Primitives

Drawing Primitive	Description	Example
Text	A string of characters (UTF-8).	
Circle	A closed curve that lies at a constant distance from a center point.	
Arc	A segment of a circle.	
Point	A point with a specified width or diameter.	
Line	A line segment defined between a pair of Vertices.	
Line Strip	A contiguous series of line segments connecting an ordered set of vertices.	
Line Loop	A closed Line Strip which includes a line segment connecting the last vertex and the first vertex.	
Triangle	A filled Line Loop formed from three vertices.	
Triangle Strip	A connected series of filled triangles formed from an ordered set of vertices. The first triangle is formed from the first three vertices. Each successive triangle is formed from the last two vertices used and the next vertex in the set.	 Note: Black outlines are for illustrative purposes only.

Drawing Primitive	Description	Example
Triangle Fan	A connected series of filled triangles formed from an ordered set of vertices. The first triangle is formed from the first three vertices. Each successive triangle is formed from the first vertex, the last vertex used, and the next vertex in the set.	 Note: Black outlines are for illustrative purposes only.

Note that each symbol definition shall include only one type of primitive. Compound symbols comprised of multiple primitive types shall be created from more than one symbol definition, each with a unique symbol ID. These symbols may then be grouped together hierarchically and controlled through the top-level, or root, symbol.

Once a symbol has been defined, the Host may add, remove, or modify primitives by sending one or more symbol definition packets. A symbol's type shall not be changed, however, unless the symbol is first destroyed and then re-defined.

5.3.3 Symbol Templates

The IG may create one or more templates defining symbol geometry. These symbol templates may be instantiated with a **Symbol Clone** packet (6.1.33). Once instantiated, the new symbols are treated like any other symbol and shall be manipulated through **Symbol Control** or **Short Symbol Control** packets.

Symbol templates may be especially useful for particularly complex symbols or those that may be duplicated often.

5.4 Coordinate Systems

5.4.1 Geodetic Coordinate System

CIGI 4 specifies a top-level (non-child) entity's position in geodetic coordinates. The coordinates define the position of the entity's reference point, typically the center of gravity. Orientation is specified by yaw, pitch, and roll angles. Sections 5.4.1.1 and 5.4.1.2 describe geodetic position and orientation, respectively.

The default Earth Reference Model (ERM) for CIGI 4 is WGS 84. The IG shall load the World Geodetic System (WGS) 84 ellipsoid at initialization. The Host may define a new reference ellipsoid by sending an **Earth Reference Model Definition** packet (Section 6.1.19) to the IG.

5.4.1.1 Position

The geodetic coordinate system uses an ellipsoidal earth model and specifies a location in terms of latitude, longitude, and altitude as shown in Figure 17.

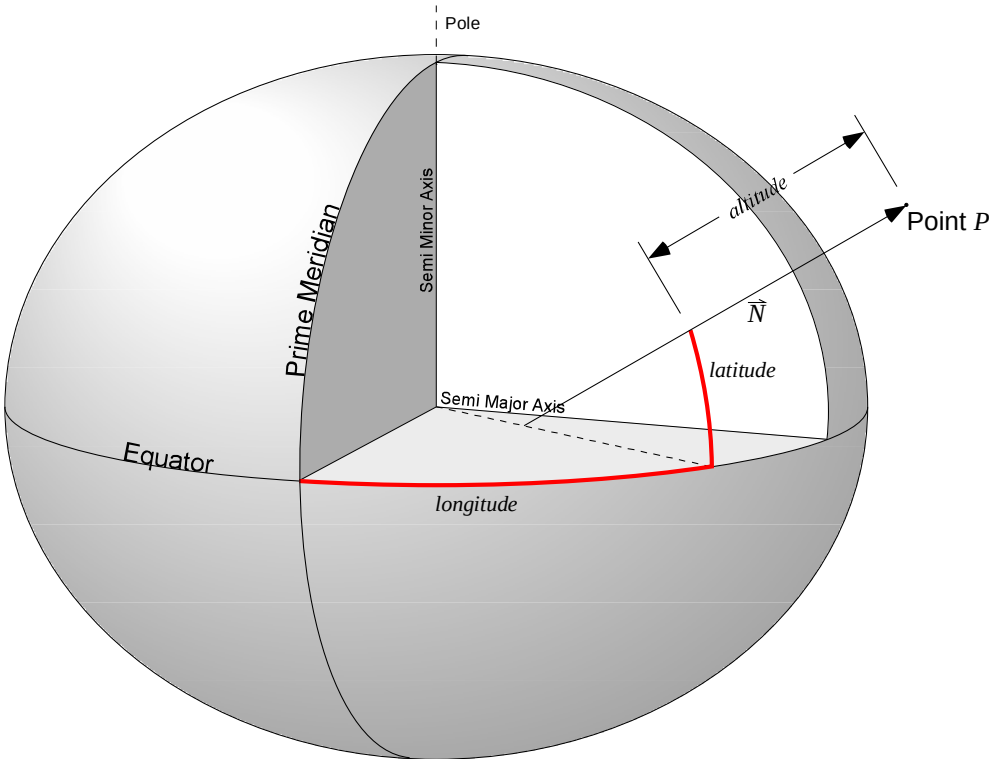


Figure 17 – Position within Geodetic Coordinate System

Given point P, an imaginary vector $\vec{N}$ extends through that point that intersects the equatorial plane and is normal to the ellipsoid's surface. *Latitude* is the size of the angle formed between this vector and the equatorial plane. This is measured in degrees north (positive) or south (negative) of the Equator and is limited to ±90°.

Longitude is the angle of the arc along the ellipsoid surface from the Prime Meridian to the point on the Equator closest to point P. This is measured in degrees east (positive) or west (negative) of the Prime Meridian and is limited to ±180°.

Altitude is the distance from P to the point of intersection between the ellipsoid surface and the normal vector. This distance is measured in meters above Mean Sea Level (MSL), which CIGI defines as the

reference ellipsoid surface. Negative values indicate that the point is below MSL, or inside the reference ellipsoid.

Note that positive *Altitude* values correspond to negative **Z** values in a North-East-Down (NED) Cartesian coordinate system, which is described in Section 5.4.1.2.

5.4.1.2 Orientation

The orientation of an entity, view, or other object with respect to the geodetic coordinate system is specified relative to a reference plane that is parallel to an ellipsoid-tangential plane. The reference plane passes through the entity's reference point. The platform-based Spatial Reference Frame (SRF) shall be a right-hand coordinate system defined so that the **X**, **Y**, and **Z** axes correspond to North, East, and down (toward the ellipsoid), respectively. This is called a North-East-Down (NED) Cartesian coordinate system or Lococentric Euclidean 3D SRF (SEDRIS, 2009) and is shown in Figure 18.

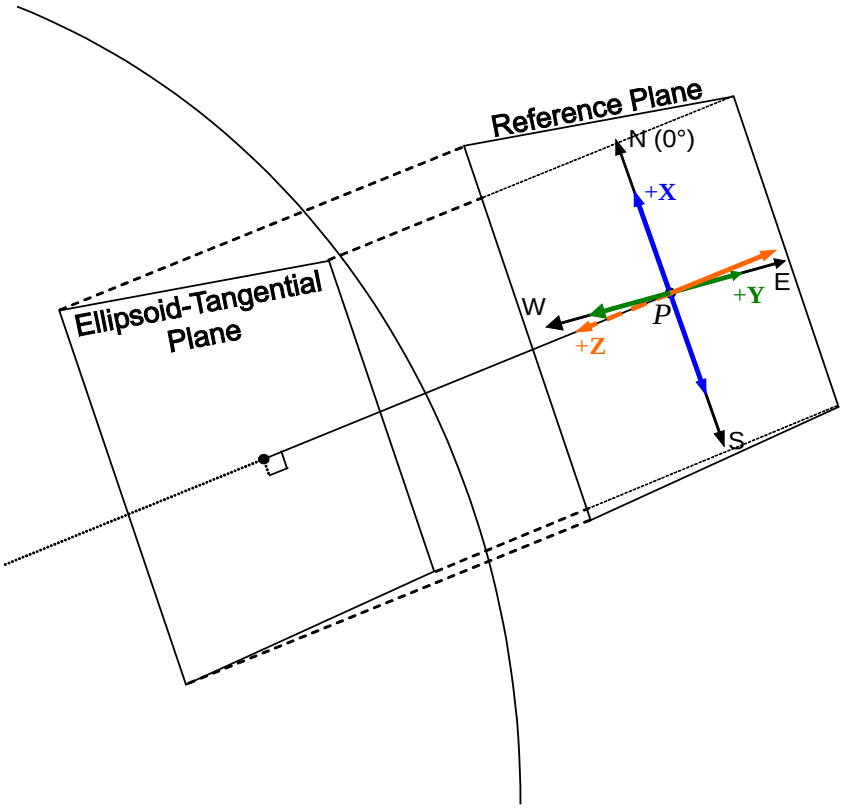


Figure 18 – Local Geodetic Reference Plane with NED Coordinate System

The order of rotation shall be about the **Z**, **Y**, and then **X** axes (i.e.,, yaw, pitch, and then roll) as illustrated below in Figure 19. This discussion assumes an entity but also applies to views, submodels, and other objects.

An entity's yaw angle, ψ, shall be measured from True North to the entity's +**X** axis. This angle shall be specified in degrees and shall increase clockwise if looking along the +**Z** axis.

An entity's pitch angle, θ, shall be measured from the reference plane to the entity's +**X** axis. This angle shall be specified in degrees positive above the reference plane, i.e., away from the ellipsoid.

Roll, φ, shall be measured from the reference plane to the entity's +**Y** axis along a plane perpendicular to the entity's **X** axis. In other words, roll is the angle of rotation about the **X** axis *after yaw and pitch have*

been applied. This angle shall be specified in degrees and shall increase clockwise from the point of view of looking along the +**X** axis.

Figure 19 illustrates the rotation about each axis:

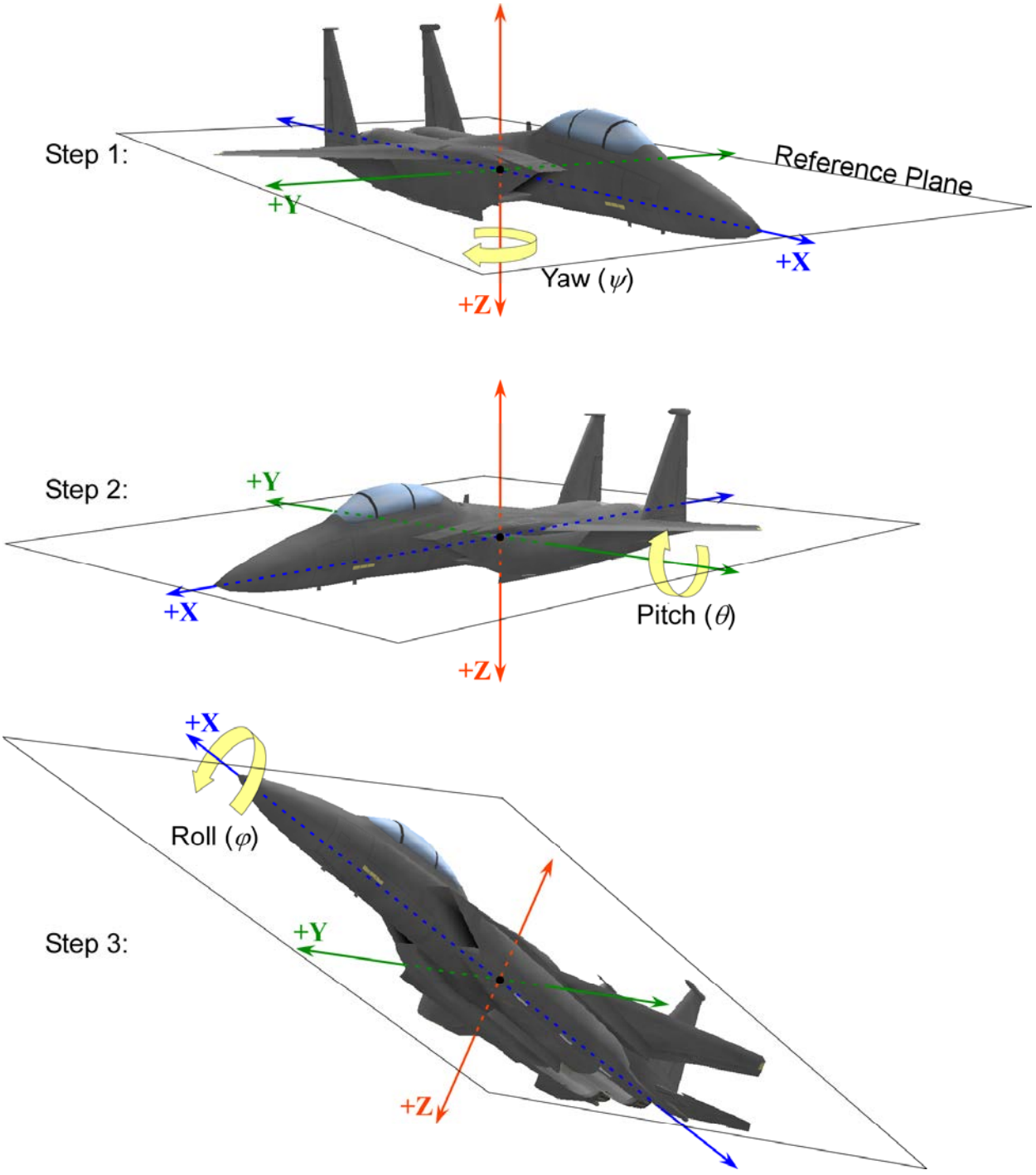


Figure 19 – Entity Rotation in NED Coordinate System

5.4.2 Entity Coordinate System

Each entity has a local NED right hand coordinate system as shown below in Figure 20. The origin corresponds to the entity's reference point, typically the center of gravity. The +**X** axis shall extend out the "front" of the entity (i.e., the nose of an aircraft). The +**Y** axis shall extend out the right side, and the +**Z** axis shall extend out the bottom of the entity.

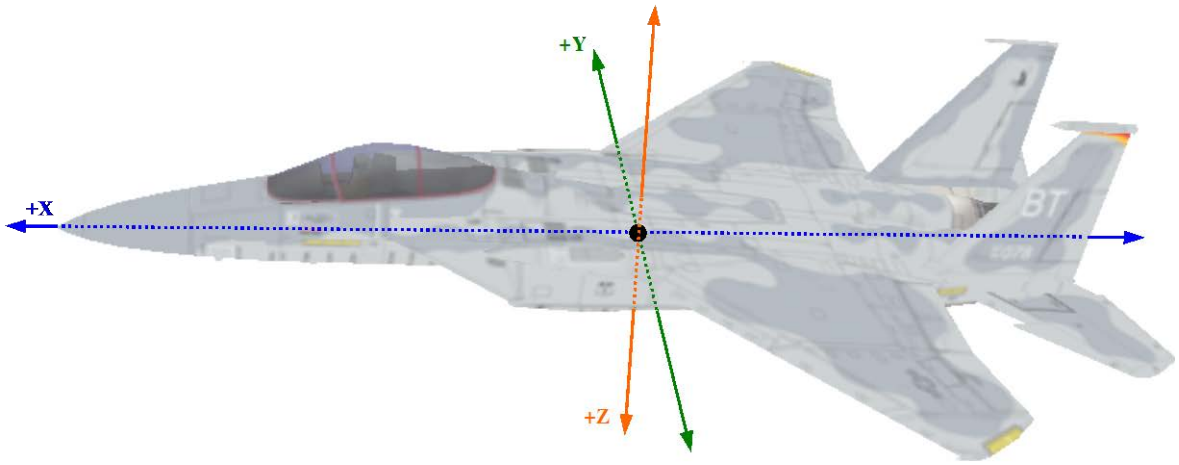


Figure 20 – Local Entity Coordinate System

An entity's local coordinate system is sometimes called its "body" coordinate system.

5.4.2.1 Position

Position with respect to an entity's local coordinate system shall be specified as the distance in meters from the entity's reference point along the entity's **X**, **Y**, and **Z** axes. The entity's local coordinate system is shown in Figure 20.

5.4.2.2 Orientation

An entity shall be rotated with respect to its reference coordinate system and shall be specified as yaw, pitch, and roll relative to a local reference plane. This reference plane is parallel to the entity's **XY** plane and passes through the local origin. The order of rotation is Yaw, Pitch, Roll as shown in Figure 19.

If the Host specifies an entity's orientation, that orientation shall override the entity's previous orientation. In other words, rotations translations applied to an entity are not cumulative.

5.4.3 Submodel Coordinate Systems

A submodel is a hierarchy of geometry nodes within a model (entity) for which a coordinate system is defined. Position and rotation of submodels are defined with respect to this coordinate system. Transformations performed on the coordinate system affect the submodel geometry as a whole. The order of rotation is Yaw, Pitch, Roll as shown in Figure 19.

The submodel coordinate system may be defined with an arbitrary position and orientation relative to the entity model's coordinate system in a way that makes sense for the submodel. For example, a leading-edge flap might have a submodel coordinate system defined as shown in Figure 21a so that applying a positive pitch angle will rotate the flap above the wing. A trailing-edge flap's submodel coordinate system, however, might need to be rotated to achieve a positive pitch above the wing as shown in Figure 21b.

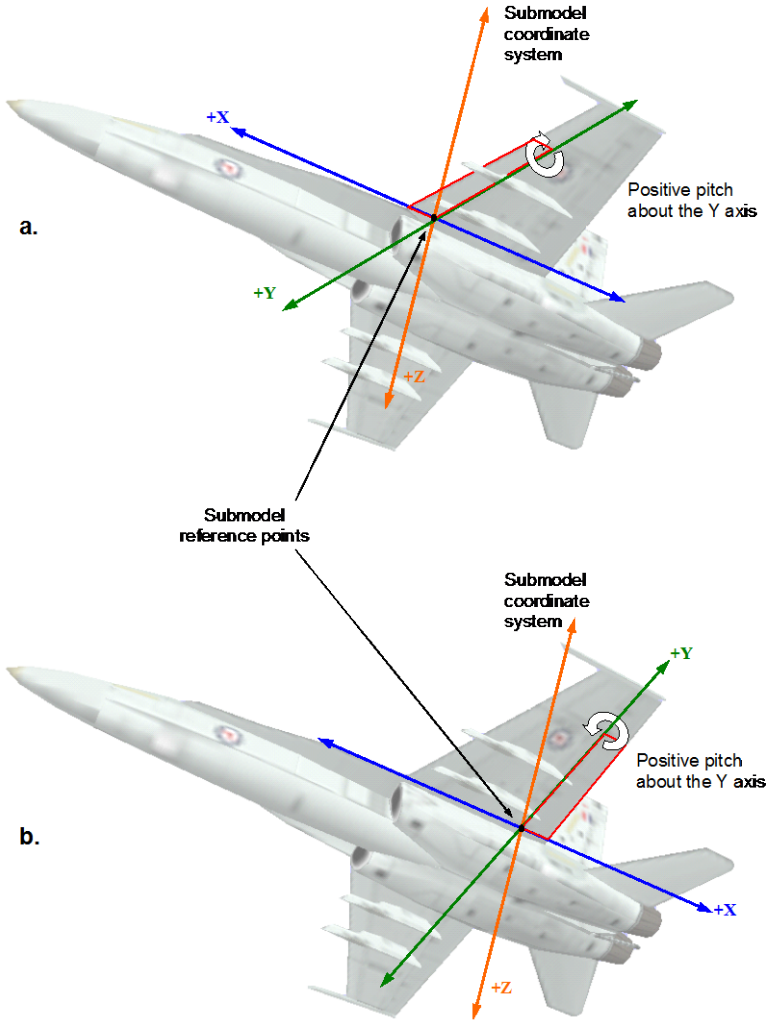


Figure 21 – Examples of Submodel Coordinate Systems

Note: Regardless of its orientation, the submodel coordinate system shall be a right-handed coordinate system.

Specifying a submodel's orientation or position shall override its previous values. In other words, rotations and translations applied to a submodel are not cumulative.

Section 6.1.6 describes the use of the **Articulated Part Control** packet in manipulating submodels.

5.4.4 Symbol Surface Placement Coordinate Systems

CIGI defines three coordinate systems that may be used to position, orient, and size symbol surfaces. These are described below.

5.4.4.1 Entity Coordinate System

A symbol surface that is attached to an entity and is not a billboard shall use the entity's local coordinate system to specify position and orientation relative to that entity as described in Section 5.4.2.

Figure 22 illustrates the positioning of the symbol surface with respect to an entity coordinate system. Note that the symbol surface's reference point shall be at the center of the surface.

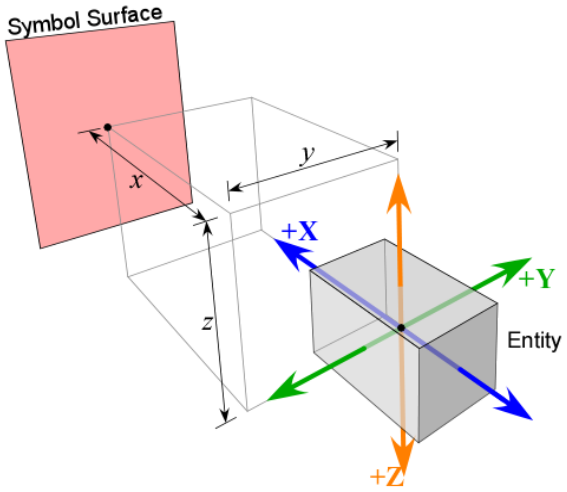


Figure 22 – Entity Coordinate System for Placing Symbol Surfaces

The translated symbol surface has a reference 3D coordinate system whose axes are parallel to those of the parent entity as shown in Figure 23:

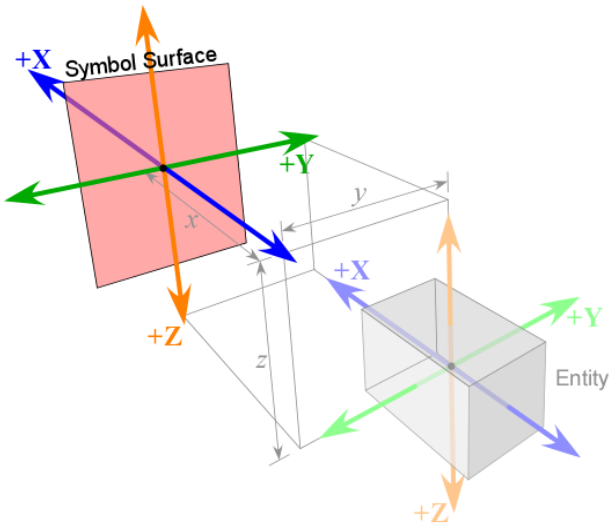


Figure 23 – Entity-Attached, Non-Billboard Symbol Surface Reference Coordinate System

This reference coordinate system is used to specify the size and rotation of the symbol surface. The surface's width shall be measured along the reference **Y** axis, and its height shall be measured along the

reference **Z** axis. Yaw, pitch, and roll rotations shall be about the **Z**, **Y**, and **X** axes, respectively and shall be in that order, as shown in Figure 24:

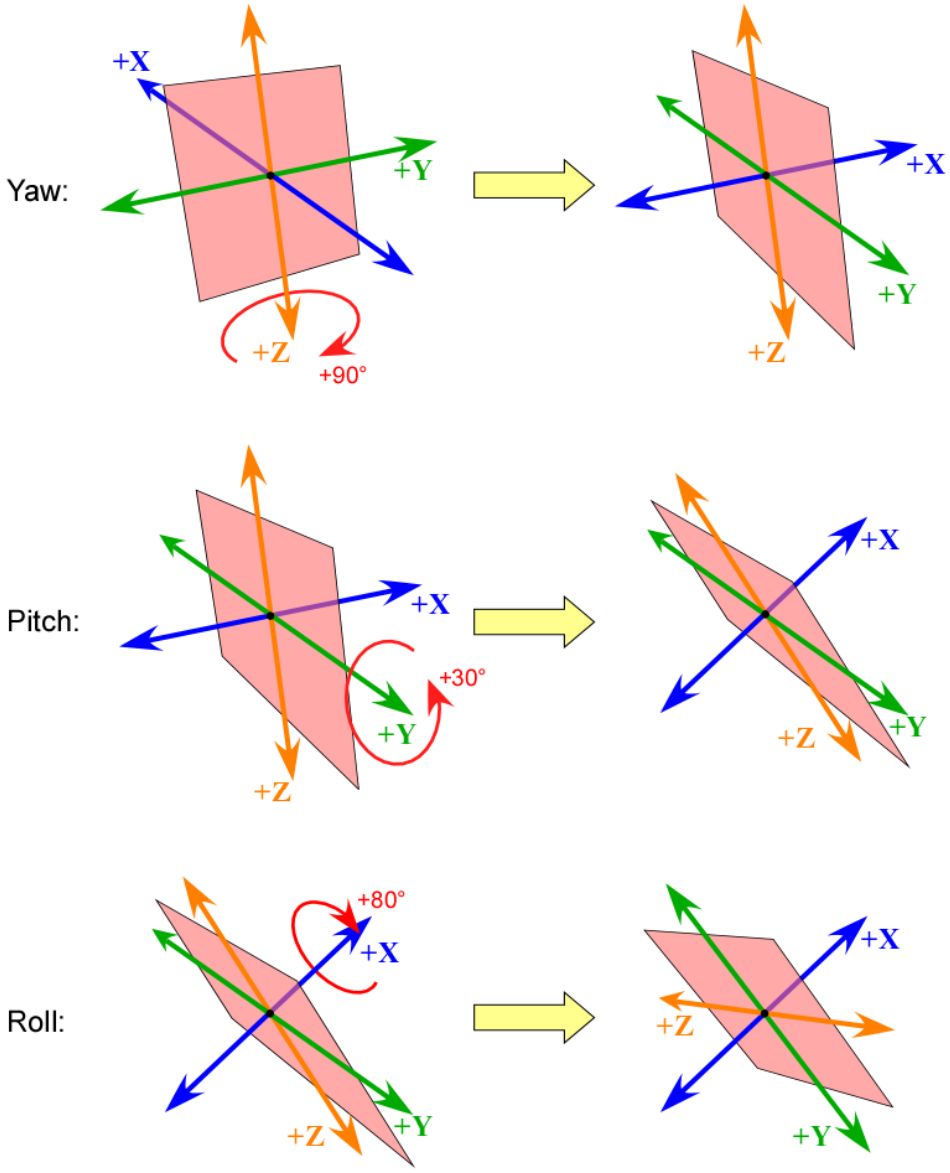


Figure 24 – Rotation of an Entity-Attached Symbol Surface

In the above example, first a yaw rotation of positive 90° is applied. This rotation is about the **Z** axis. Next, a pitch angle of positive 30° is applied by rotating the surface about the **Y** axis. Finally, the surface is rolled about the **X** axis through an angle of positive 80°.

Note that the method and order of rotation is the same as that described for child entities in Section 5.4.2.2.

5.4.4.2 Symbol Surface Billboard Coordinate System

A symbol surface that is attached to an entity and is a billboard shall be on a plane parallel to the view's viewport. A normal vector emanating from the center of the symbol surface toward the eyepoint shall be parallel to, and in the opposite direction of, the view's viewing vector. Thus, the surface's **U** axis (see

Section 5.4.5.1) will be parallel to the near clipping plane's bottom and top sides, while the surface's **V** axis will be parallel to the near clipping plane's left and right sides.

The center of the symbol surface shall be specified relative to the entity. However, since the surface's orientation and position are independent of the entity's yaw, pitch, and roll, the symbol surface is not able to be placed with respect to the entity's local coordinate system. Rather, the position is specified with respect to a three-dimensional, right-handed coordinate system whose origin is at the center of the symbol surface. The **Z** axis shall extend along the aforementioned normal vector and shall be positive in the direction toward the eyepoint. The **X** axis shall be horizontal relative to the view and shall be positive to the right from the perspective of the eyepoint. Likewise, the **Y** axis shall be vertical relative to the view and shall be positive in the up direction from the perspective of the eyepoint.

Figure 25 illustrates the placement of a billboard surface relative to an entity and the viewport.

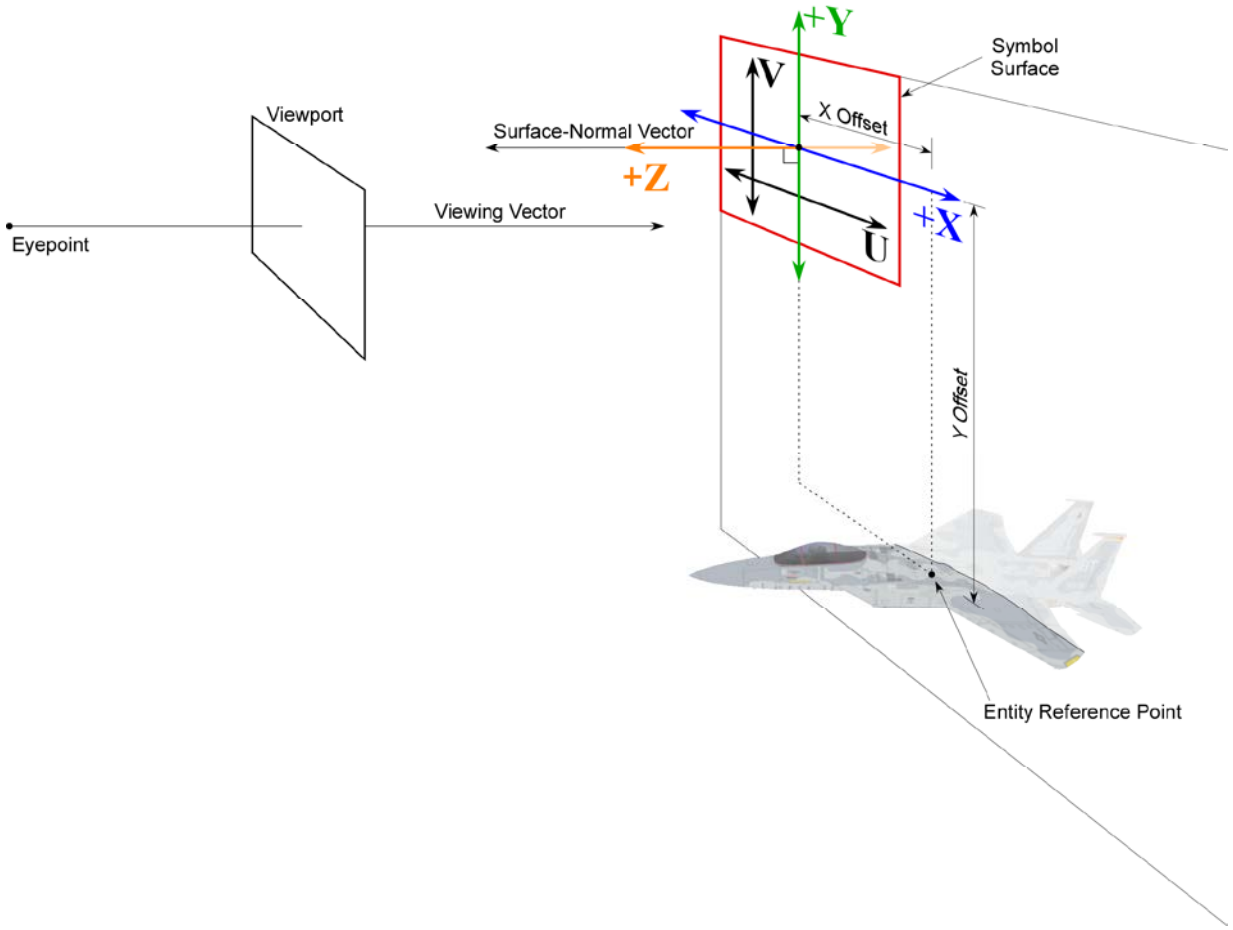


Figure 25 – Symbol Surface Billboard Coordinate System

The above illustration shows a symbol surface with an arbitrary UV origin. The XYZ origin is at the center of the surface and is not necessarily co-located with the UV origin. The **X** axis is parallel to the top and bottom of the viewport, and the **Y** axis is parallel to the left and right sides of the viewport. The surface's position is specified by X, Y, and Z offsets from the entity's reference point to the symbol surface's reference point. The Z offset in this example is zero, so the entity's reference point is coplanar with the surface.

The width and height of the symbol surface are defined along the surface's **X** and **Y** axes, respectively.

5.4.4.3 Normalized Viewport Coordinate System

A symbol surface that is attached to a view is overlaid onto the view's viewport. The size and position of the symbol surface shall be defined in terms of the width and height of the viewport. The coordinates (0, 0) and (1.0, 1.0) shall be mapped to the lower-left and upper-right corners of the viewport, respectively. The symbol surface shall be placed with respect to the 2D Cartesian coordinate system implied by those points as shown in Figure 26:

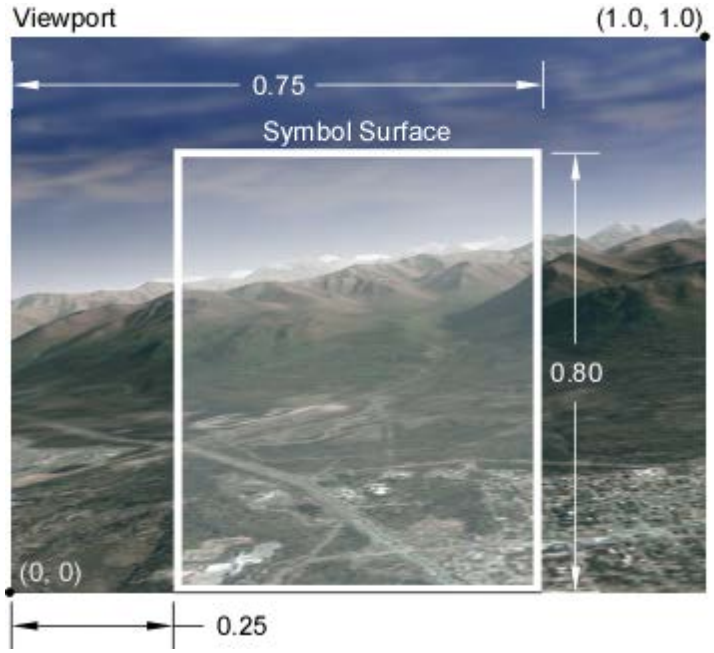


Figure 26 – Example of a View-Attached Symbol Surface

In the example above, the symbol surface is defined with left and right sides at 0.25 and 0.75 horizontal viewport units from the origin, respectively. The bottom is placed along the lower viewport boundary, or at 0.0, and the top is placed at 0.80 vertical units.

By defining the size and position of the symbol surface in this manner, they remain constant for a given viewport even if the half-angles defining the viewing volume change.

5.4.5 Symbol Placement Coordinate Systems

After a symbol surface has been defined and placed in either 3D virtual space or in viewport space, symbols may be drawn on the surface. The position, size, and orientation of symbols shall be defined with respect to the symbol surface's 2D coordinate system or the parent symbol's local 2D coordinate system as described below.

5.4.5.1 Symbol Surface 2D Coordinate System

A symbol surface is associated with a 2D Cartesian coordinate system (UV mapping) that is used to define the sizes of symbols and to position top-level symbols drawn on the surface. The horizontal axis, or **U** axis, is parallel to the top and bottom boundaries of the surface. The vertical, or **V**, axis is parallel to the left and right boundaries of the surface. When defining the symbol surface, the Host specifies the u and v values that correspond to the left, right, top, and bottom boundaries of the surface. Figure 27 continues the example from Figure 26, depicting a UV coordinate system applied to the symbol surface:

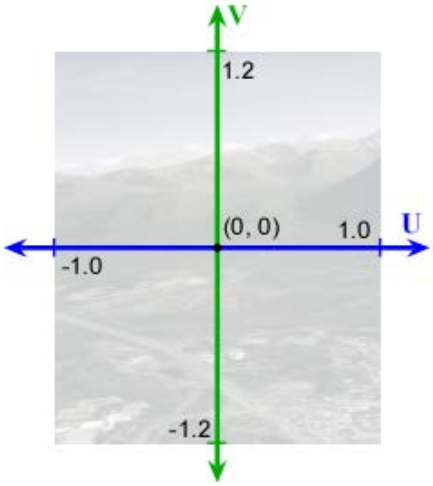


Figure 27 – Example of a Symbol Surface 2D Coordinate System

Assuming that the size of the view is 1600×1200 pixels, the aspect ratio of the symbol surface would be 1.2:1 as shown:

$$\frac{1200}{1600} \cdot \frac{0.80}{(0.25 - 0.75)} = \frac{1.2}{1}$$

The symbol surface's horizontal and vertical extents have been chosen to be ±1.0 and ±1.2 units, respectively, to match this aspect ratio. This produces units that are "square" (assuming square pixels) and defines the origin to be at the center of the surface.

Figure 28 shows another example of how a UV coordinate system may be defined on the same symbol surface. The origin is at the lower-left corner of the symbol surface. The width and height of the surface correspond to its dimensions in screen space (i.e.,, in pixels). This provides a one-to-one mapping of UV tuples to pixels and allows for pixel- or subpixel-accurate manipulation of symbol primitives.

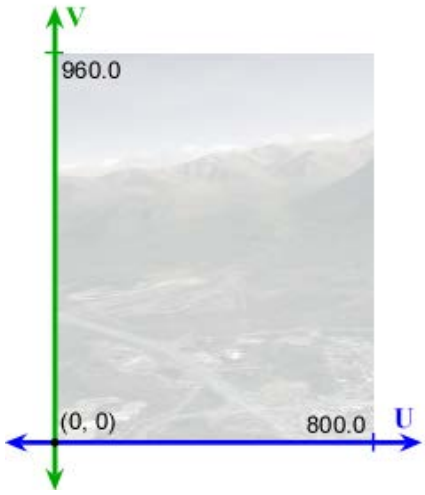


Figure 28 – Example #2 of a Symbol Surface 2D Coordinate System

Note that the origin of a UV coordinate system may be placed outside the bounds of the symbol surface. The only requirement in defining the UV coordinate system is that the left side shall be defined to be less than the right side and that the bottom shall be defined to be less than the top.

A symbol's primitives' dimensions shall be specified in units defined by the symbol surface's UV coordinate system. For example, if a circle primitive is defined with an outer radius of 10, then the circle extends 20 horizontal units along the **U** axis and 20 vertical units along the **V** axis. If the horizontal and vertical units are not the same length, then the circle primitive shall be drawn as an ellipse.

The symbol as a whole may be scaled along its **U** and **V** axes. The horizontal and vertical units used by the symbol and its children shall be affected by these scaling factors; thus, lengths and distances shall be measured in *scaled symbol surface units*.

Note that scaling operations shall be performed after the reference coordinate system is first translated and then rotated.

5.4.5.2 Symbol 2D Coordinate System

Every symbol has a local Cartesian coordinate system. This coordinate system shall be used to rotate the symbol and to position the symbol's primitives and child symbols. The origin of this coordinate system shall correspond to the symbol's reference point.

Figure 29 shows a top-level symbol (an arc) and its local coordinate system in relation to the symbol surface:

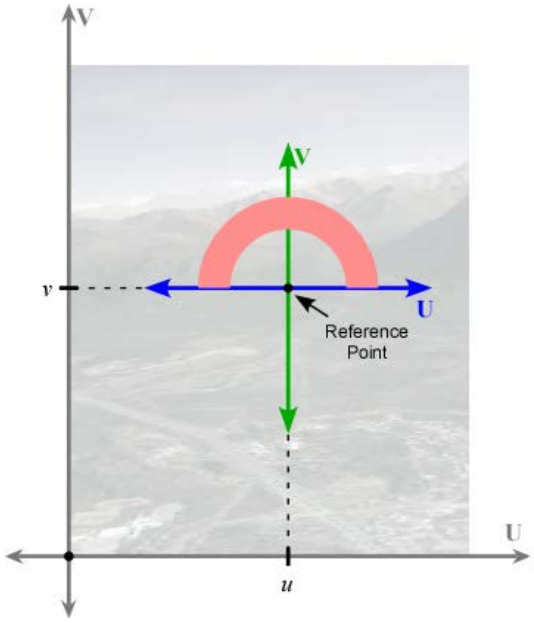


Figure 29 – Symbol 2D Coordinate System Relative to a Symbol Surface

The origin of the top-level symbol's local coordinate system, as well as the symbol's reference point, are located at some point (u, v) in the symbol surface's UV coordinate system. Before any transformation is applied to the symbol, its local **U** and **V** axes are parallel to the symbol surface's **U** and **V** axes and the horizontal and vertical units for both sets of axes are equal in length.

Two or more symbols may be arranged in a hierarchy. A child symbol's local coordinate system shall be specified relative to the immediate parent's coordinate system. Figure 30 illustrates a child symbol's coordinate system in relation to the parent's coordinate system.

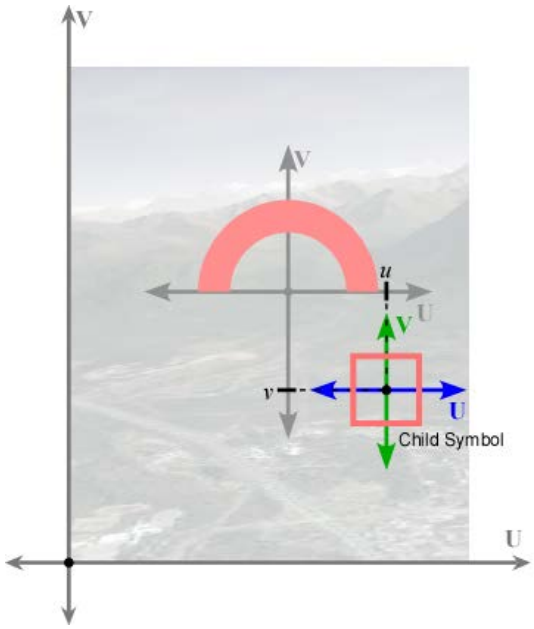


Figure 30 – Child Symbol Coordinate System Relative to Parent

The direction of rotation of a symbol about its reference point shall be counter-clockwise. When a symbol is rotated, the symbol's local coordinate system shall also be rotated. As a result, any child symbols shall be rotated with the parent as shown in Figure 31.

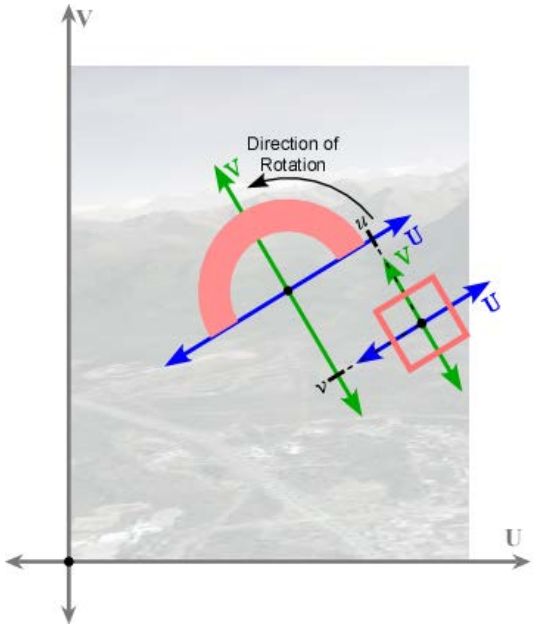


Figure 31 – Rotated Parent Symbol with Child

Scaling shall be specified independently for a symbol's local **U** and **V** axes. Both scalars are positive real numbers. The following example shows the effect of scaling a symbol. Scaling is illustrated only along the **U** axis for clarity.

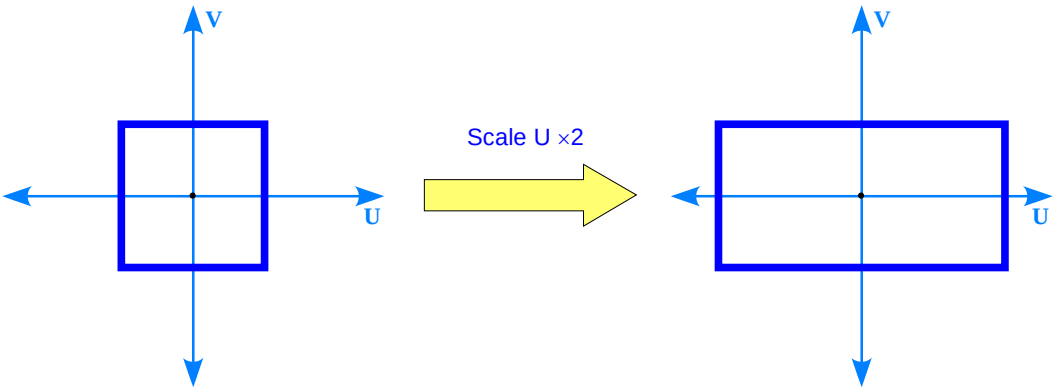


Figure 32 – Scaling a Symbol

Figure 33 shows the same symbol overlapped with a child symbol rotated 45°. The figure shows the result of scaling only the parent.

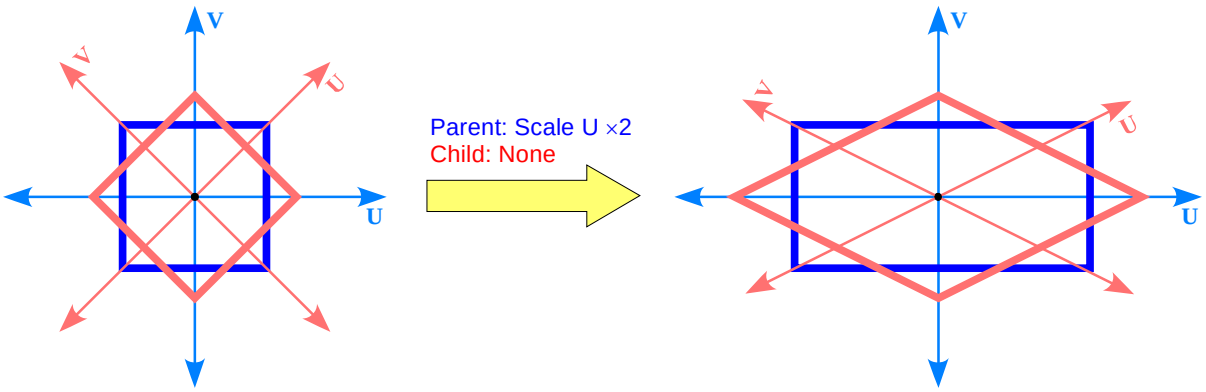


Figure 33 – Scaling a Parent Symbol

Figure 34 shows the result of scaling only the child.

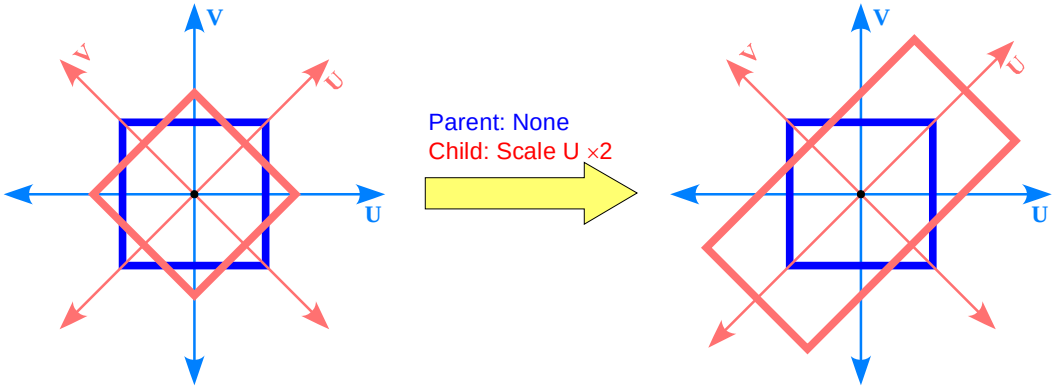


Figure 34 – Scaling a Child Symbol

Figure 35 shows the result of scaling both the parent and the child.

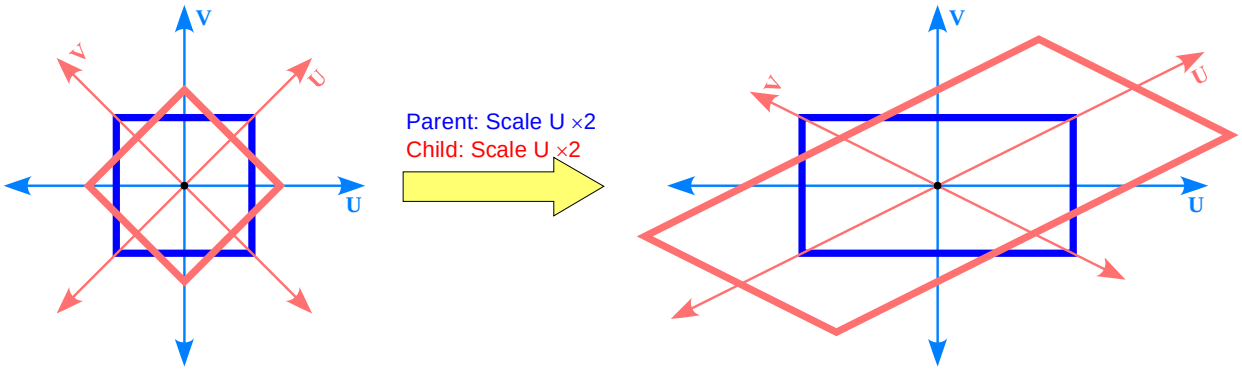


Figure 35 – Scaling a Parent Symbol and a Child Symbol

Scaling factors applied to any given local coordinate system shall not be cumulative. In other words, when a scaling factor is specified for a symbol along one of its axes, any scaling factor previously defined for that axis of that symbol shall be discarded. The appearance of the symbol shall still be affected by the parent's scale, however.

6. Data Packet Reference

This portion of the ICD describes each packet in the Common Image Generator Interface. Section 6.1 describes the CIGI packets used in Host-to-IG messages. Section 6.2 describes the packets used in IG-to-Host messages. Section 6.3 discusses user-defined packets, which may be used for either Host-to-IG or IG-to-Host messages.

Each topic contains one or more paragraphs describing the use of the packet. Following this text is a diagram illustrating the structure of the packet. An example of such a diagram is shown in Figure 36.

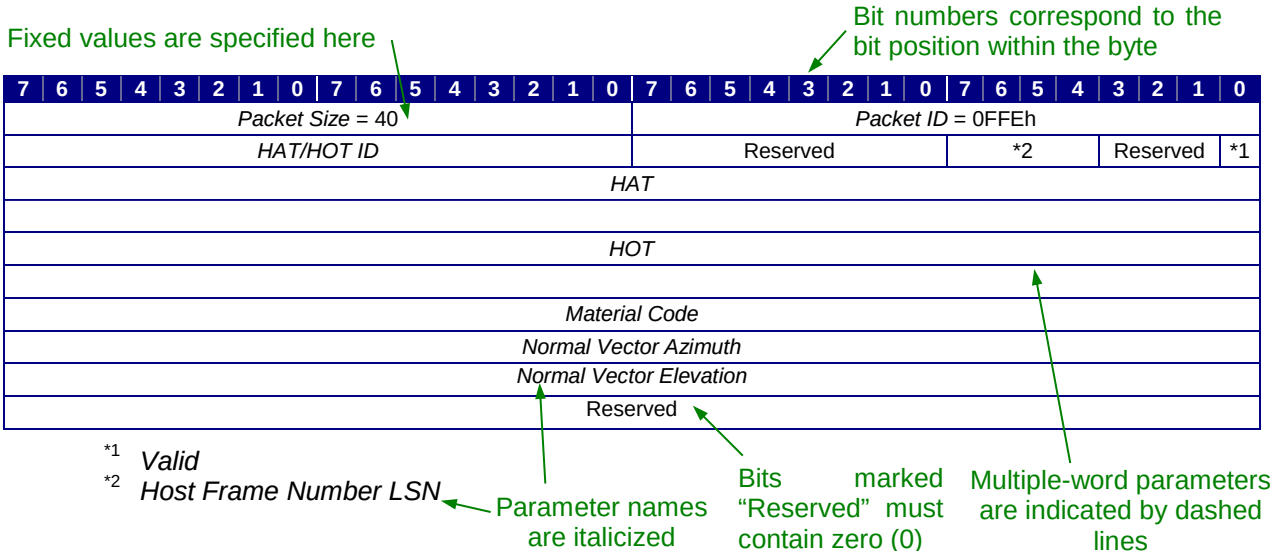


Figure 36 – Example of Packet Structure Diagram

Each row represents a 32-bit word. The topmost row corresponds to the first word in the packet; the memory address increases downward. 64-bit parameters are indicated by a dotted line representing the enclosed word boundary. The parameter name appears in the first of the two rows.

Parameter names are italicized. Parameters whose names do not fit within the space allotted in the table are noted below the diagram. Bits marked "Reserved" are allocated for future use and shall be populated with zeros (0).

Bit positions are numbered from right to left. The leftmost bit in each byte is the most significant bit. The leftmost byte in each word has the lowest physical address.

The following illustration shows how the above packet would be stored in memory on both a big-endian and a little-endian computer:

Little Endian		Big Endian
Packet Size (LSB)	**Low Address**	*Packet Size (MSB)*
Packet Size (MSB)		*Packet Size (LSB)*
Packet ID (LSB)		*Packet ID (MSB)*
Packet ID (MSB)		*Packet ID (LSB)*
HAT/HOT ID (LSB)		*HAT/HOT ID* (MSB)
HAT/HOT ID (MSB)		*HAT/HOT ID* (LSB)
0		0
Valid/Frame Number LSN		*Valid/Frame Number LSN*
HAT (LSB)		*HAT* (MSB)
HAT (2nd-order byte)		*HAT* (7th-order byte)
HAT (3rd-order byte)		*HAT* (6th-order byte)
HAT (4th-order byte)		*HAT* (5th-order byte)
HAT (5th-order byte)		*HAT* (4th-order byte)
HAT (6th-order byte)		*HAT* (3rd-order byte)
HAT (7th-order byte)		*HAT* (2nd-order byte)
HAT (MSB)		*HAT* (LSB)
HOT (LSB)		*HOT* (MSB)
HOT (2nd-order byte)		*HOT* (7th-order byte)
HOT (3rd-order byte)		*HOT* (6th-order byte)
HOT (4th-order byte)		*HOT* (5th-order byte)
HOT (5th-order byte)		*HOT* (4th-order byte)
HOT (6th-order byte)		*HOT* (3rd-order byte)
HOT (7th-order byte)		*HOT* (2nd-order byte)
HOT (MSB)		*HOT* (LSB)
Material Code (LSB)		*Material Code* (MSB)
Material Code (2nd-order byte)		*Material Code* (3rd-order byte)
Material Code (3rd-order byte)		*Material Code* (2nd-order byte)
Material Code (MSB)		*Material Code* (LSB)
Normal Vector Azimuth (LSB)		*Normal Vector Azimuth* (MSB)
Normal Vector Azimuth (2nd-order byte)		*Normal Vector Azimuth* (3rd-order byte)
Normal Vector Azimuth (3rd-order byte)		*Normal Vector Azimuth* (2nd-order byte)
Normal Vector Azimuth (MSB)		*Normal Vector Azimuth* (LSB)
Normal Vector Elevation (LSB)		*Normal Vector Elevation* (MSB)
Normal Vector Elevation (2nd-order byte)		*Normal Vector Elevation* (3rd-order byte)
Normal Vector Elevation (3rd-order byte)		*Normal Vector Elevation* (2nd-order byte)
Normal Vector Elevation (MSB)		*Normal Vector Elevation* (LSB)
0		0
0		0
0		0
0	**High Address**	0

Figure 37 – Example of Packet Data Storage

Below the packet structure diagram is a table that lists each parameter and describes its use. This table identifies the parameter's name, data type, and unit of measure. If a parameter's values differ from those listed for the data type in Table 2 on page 31, those values are listed next. If applicable, a default value and reference datum are listed next.

Table 4 describes the general format of a packet parameter definitions table:

Table 4 – Format of Packet Parameter Definitions Table

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this type of data packet. The value 40 in this example specifies the number of bytes in a **HAT/HOT Extended Response** packet.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFEh	This parameter specifies the packet's identifier, which uniquely identifies the packet. In this example, the value 0FFEh indicates that this is a **HAT/HOT Extended Response** packet. This is a dimensionless value.
Parameter Name Type: double float Units: degrees Values: -90.0 – 90.0 Default: 0.0 Datum: Equator	The remaining items in the table describe the rest of the packet's parameters. There is one row per parameter, and the parameters are given in the order in which they appear in the packet (left to right). The parameter is identified by the parameter name. Below this name is the data type. CIGI data types are described in Section 4.6. Next is the unit of measure used for the parameter. If the parameter is dimensionless and has no unit, this is indicated by "N/A." The domain may be specified below the unit of measure. This might either be a range of values or an enumerated list of discrete values. If no values are specified, then the domain is the entire range of the data format as listed in Table 2. Below the domain is default value, if applicable, for the parameter. This is the value assigned to the specified attribute during initialization of the IG. Attributes of entities and other objects that are instantiated at runtime will not have a default value. Finally, a reference datum may be specified for the attribute. This is the point or state from which the value is measured.

6.1 Host-to-IG Packets

6.1.1 IG Control

The **IG Control** packet is used to control the IG's operational mode, database loading, and timing correction. This shall be the *first* packet in each Host-to-IG message, and every Host-to-IG message shall contain *exactly one* **IG Control** packet. If more than one is encountered during a given frame, the resulting IG behavior will be undefined.

The **IG Control** packet allows the Host to control the loading of terrain. Each database is associated with a number from 1 to 127. The Host shall set the *Database Number* parameter to the appropriate value to direct the IG to begin reading the corresponding database into memory. An example is shown in Figure 38:

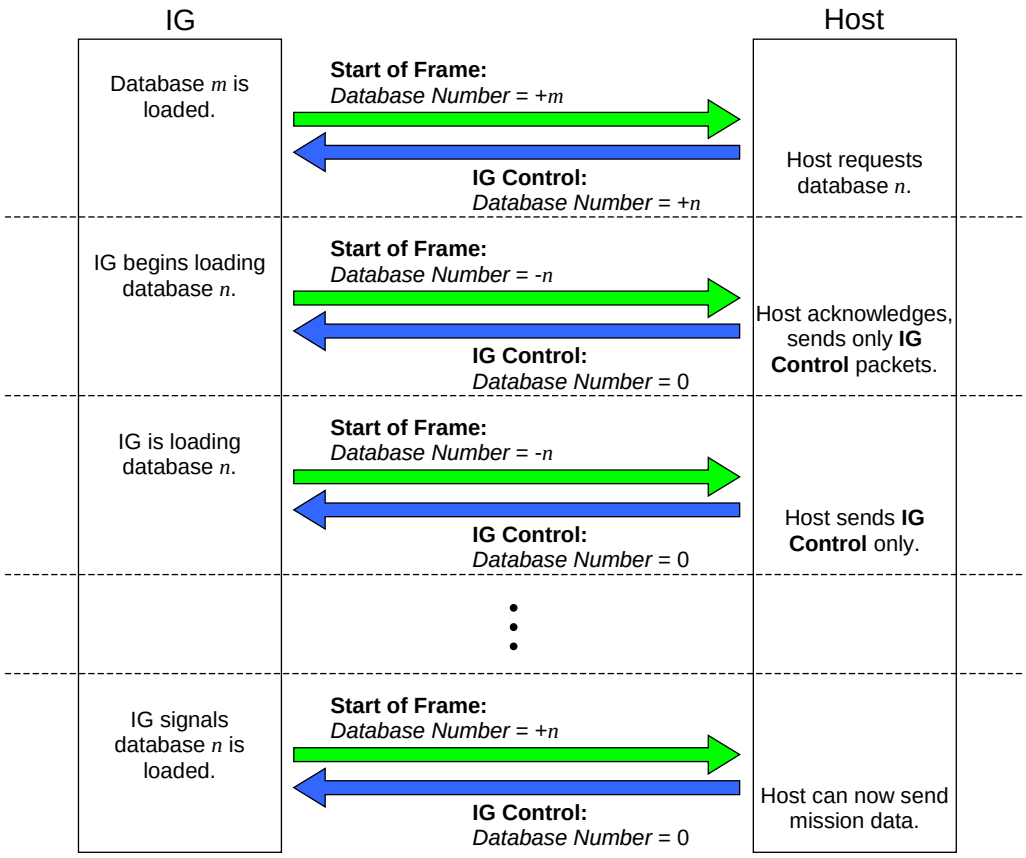


Figure 38 – Database Loading Sequence in Synchronous Mode

The IG shall indicate that the database is being loaded by negating the value and placing it in the *Database Number* parameter of the **Start of Frame** packet. The Host shall then acknowledge this change by setting the *Database Number* parameter of the **IG Control** packet to zero (0). Because the IG's resources may be devoted to disk I/O and other functions, the Host shall send only **IG Control** packets at this time.

After the IG receives the acknowledgement, it shall signal the completion of the database load by setting the *Database Number* parameter of the **Start of Frame** packet to the positive database number. The IG shall then process mission data from the Host.

Note that the IG shall ignore the *Database Number* parameter while in Reset/Standby mode.

When exclusively using a single database predefined on the IG to load at startup, the IG shall set the *Database Number* of the **Start of Frame** packet to zero (0). When the Host detects a zero in this parameter, it shall in turn set the *Database Number* parameter of the **IG Control** packet to zero (0).

The contents of the **IG Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 24</td><td colspan="16">Packet ID = 0h</td></tr>
<tr><td colspan="8">Major Version = 4</td><td colspan="8">Database Number</td><td colspan="8">Reserved</td><td>*4</td><td colspan="3">Minor Version</td><td>*3</td><td>*2</td><td>*1</td></tr>
<tr><td colspan="32">Host Frame Number</td></tr>
<tr><td colspan="32">Last IG Frame Number</td></tr>
<tr><td colspan="32">Timestamp</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *IG Mode*
*2 *Timestamp Valid*
*3 *Smoothing Enable*
*4 *Entity Type Substitution Enable*

Figure 39 – IG Control Packet Structure

Table 5 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 5 – IG Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 0h	This parameter identifies this data packet as the **IG Control** packet. The Host shall set this parameter to 0h.
Major Version Type: unsigned int8 Units: N/A Value: **4**	This parameter indicates the major version of the CIGI interface that is currently being used by the Host. The IG should use this number to determine concurrency. The Host shall set the value of this parameter to **4**.

Parameter	Description
Database Number Type: int8 Units: N/A Values: 0 No load requested 1 – 127 Identifies desired database Default: 0	This parameter is used to initiate a database load on the IG. If the Host sets this parameter to a non-zero value, the IG shall begin loading the database that corresponds to that value. If the number corresponds to the current database, then the database shall be reloaded. The IG shall indicate that the database is being loaded by negating the value and placing it in the *Database Number* parameter of the **Start of Frame** packet. When the Host receives this notification, it should set the value of the *Database Number* parameter of the **IG Control** packet to zero (0) to prevent continuous reloading of the database on the IG. The IG shall ignore this parameter while in Reset/Standby mode. Refer to Section 6.2.1 for more information on the **Start of Frame** packet.
Entity Type Substitution Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether an IG is allowed to substitute a different entity type if the requested extended entity type specified in an **Entity Control** packet is not available on the IG. If the Host sets this parameter to Disable (0), then the IG shall disable the substitution of entity types for all entities. If the Host sets this parameter to Enable (1), then the IG shall enable the substitution of entity types for all entities.
Minor Version Type: unsigned int8 Units: N/A Value: **0**	This parameter indicates the minor version of the CIGI interface that is currently being used by the Host. The IG should use this number to determine concurrency. The Host shall set this parameter to **0**.
IG Mode Type: unsigned 2-bit field Units: N/A Values: 0 Reset/Standby 1 Operate 2 Debug Default: 0	The value of this parameter shall determine the IG's operational mode. The Host may initiate a mode change by setting the value of this parameter to the desired mode. When the IG completes the mode change, the IG shall set the value of the *IG Mode* parameter in the **Start of Frame** packet accordingly. For information on each of the IG modes, refer to the description of the *IG Mode* parameter of the **Start of Frame** packet in Table 48.

Parameter	Description
Timestamp Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Timestamp* parameter contains a valid value. If the Host populates the *Timestamp* parameter with an accurate clock value, then the Host should set *Timestamp Valid* to Valid (1); otherwise, the Host shall set this parameter to Invalid (0). The Host should correctly populate the *Timestamp* parameter and set *Timestamp Valid* to Valid (1) in asynchronous mode.
Smoothing Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether any "dead reckoning" or other entity extrapolation or interpolation algorithms may be used by the IG. If the Host sets this parameter to Disable (0), then the IG shall disable extrapolation or interpolation for all entities. If the Host sets this parameter to Enable (1), then the IG shall enable or disable extrapolation/interpolation per entity according to the *Smoothing Enable* flag in the **Entity Control** packet.
Host Frame Number Type: unsigned int32 Units: N/A	The value of this parameter uniquely identifies a data frame on the Host. The Host shall increment this value by one (1) for each successive message. Note: In CIGI 3.0/3.1 this parameter was named "Frame Counter" and the Host populated it with the value of the *Frame Counter* parameter from the last **Start of Frame** packet received. As of CIGI 3.2, however, the *Host Frame Number* is independent of the *IG Frame Number* parameter in the **Start of Frame** packet.
Last IG Frame Number Type: unsigned int32 Units: N/A	This parameter identifies the most recent IG-to-Host message received by the Host. The Host shall populate this parameter with the value of the *IG Frame Number* parameter in the last **Start of Frame** packet received from the IG. This parameter serves as an acknowledgement that the Host received the last message.

Parameter	Description
Timestamp Type: unsigned int32 Units: 10 microseconds (μs) Datum: Arbitrary reference time	This parameter indicates the number of 10μs "ticks" since some initial reference time. In asynchronous mode the IG should use this value to correct for latencies as described in Section 4.2.1. In synchronous mode the IG may use this value to accomplish correction to entity position and orientation when dead reckoning. The 10μs unit allows the simulation to run for approximately 12 hours before a timestamp rollover occurs. The IG software should contain logic to detect and correct for rollover. The Host shall populate this parameter with a valid time in asynchronous operation. The use of this parameter is optional for synchronous operation. If this parameter does not contain a valid timestamp, then the Host shall set the *Timestamp Valid* parameter to zero (0).

6.1.2 Entity Position

The **Entity Position** packet is used to control position, attitude, and other attributes describing an entity's state. This packet applies to all entities in the simulation.

When the Host sends an **Entity Position** packet to the IG, the IG shall set the position and attitude of the entity corresponding to the value of the *Entity ID* parameter. The IG shall disregard the data packet if the entity associated with the *Entity ID* has not been previously created by the Host having sent an **Entity Control** packet with the *Entity State* set to Inactive/Standby (0) or Active (1) to instantiate the entity.

Entities may be attached to one another in a hierarchical relationship with each child entity's position specified relative to its parent's coordinate system. When the Host repositions or rotates a parent entity, any children shall move along with the parent. No explicit manipulation of a child entity is necessary unless its position and attitude change with respect to its parent.

In the example depicted in Figure 112 on page 259 the missiles and missile trails could be maneuvered in one of two ways. First, each missile and missile trail could be controlled uniquely, requiring the host to provide position and attitude data for each of the four entities. The simpler and typically preferred way would be to establish a parent-child relationship for each missile and missile trail pair. Each missile could be attached to the aircraft until launched, at which time the missile would be detached from its parent and promoted to a top-level entity. Each top-level missile would then possess its own independent hierarchy, which would be manipulated independently from the aircraft. Figure 40 illustrates the parent-child hierarchy before and after a missile is launched:

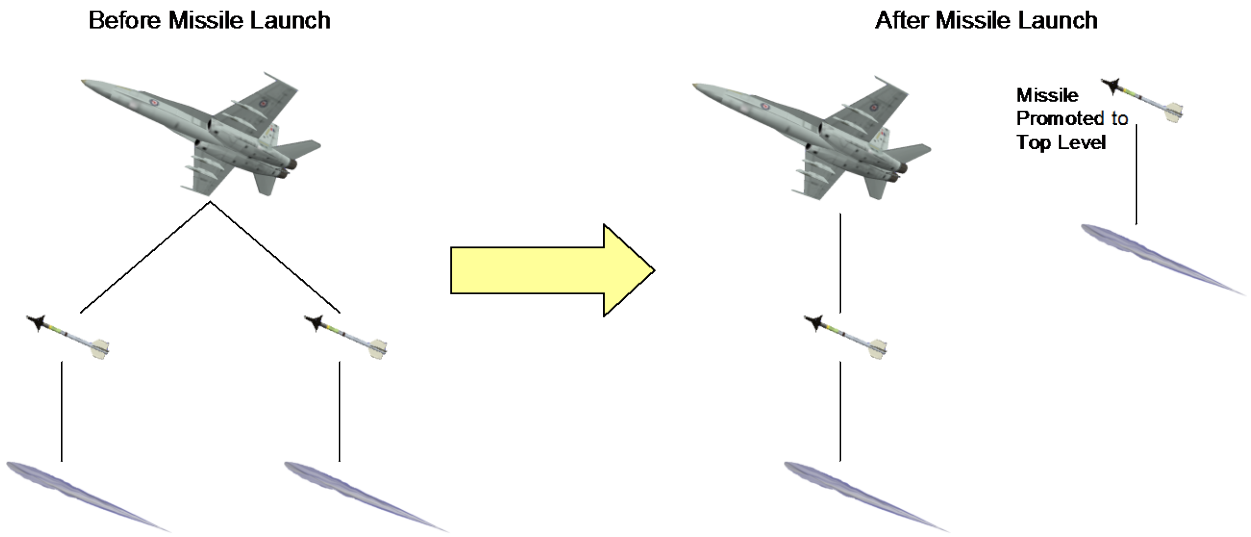


Figure 40 – Example of Child Entity Detachment

The *Attach State* parameter of the **Entity Position** packet determines whether an entity is attached to a parent. If this parameter is set to Attach (1), the entity shall be attached to the entity specified by the *Parent ID* parameter. Otherwise, the entity shall be a top-level entity.

The positions of top-level entities (i.e.,, those entities that are not children) shall be specified as a geodetic latitude, longitude, and altitude (see Section 5.4.1.1). The positions of child entities shall be specified with respect to the parents' NED body coordinate systems (see Section 5.4.2).

In certain instances, it may be desirable for the IG to "clamp" the entity to the ground or water surface. If the *Ground/Ocean Clamp* parameter is set to Non-Conformal (1) or Conformal (2), the *Altitude* parameter shall specify an offset above the ground or sea surface height. This would be useful for specifying the vertical distance from an automobile's reference point to its wheels, for instance, or from a ship's

reference point to its waterline. Similarly, *Roll* and *Pitch* shall specify rotational offsets when ground or ocean clamping is enabled.

Once an **Entity Position** packet for an entity is sent to the IG, the IG shall continue using this position and attitude in conjunction with any **Velocity Control** and **Acceleration Control** packets for the same entity ID until another **Entity Position** packet specifying that entity ID is received. For example, packets describing two aircraft may be sent every frame to indicate continuous positional changes, while a packet describing an inactive SAM site may be sent once during mission initialization.

The contents of the **Entity Position** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 48																*Packet ID* = 01h															
Reserved						*2	*1	Reserved								*Entity ID*															
Parent ID																Reserved															
Roll																															
Pitch																															
Yaw																															
Latitude/X Offset																															
Longitude/Y Offset																															
Altitude/Z Offset																															

*1 *Attach State*
*2 *Ground/Ocean Clamp*

Figure 41 – Entity Position Packet Structure

Table 6 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 6 – Entity Position Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 48.
Packet ID Type: unsigned int16 Units: N/A Value: 01h	This parameter identifies this data packet as the **Entity Position** packet. The Host shall set this parameter to 01h.

Parameter	Description
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the entity may be attached as a child to a parent. If the Host sets this parameter to Detach (0), then the entity shall become or remain a top-level (non-child) entity. The *Parent ID* parameter shall be ignored. The *Yaw*, *Pitch*, *Roll*, *Latitude*, *Longitude*, and *Altitude* parameters shall specify the entity's position relative to the geodetic coordinate system (see Section 5.4.1). If the Host sets this parameter to Attach (1), then the entity shall become or remain attached to the entity specified by the *Parent ID* parameter. The parent needs to already exist, having been created in a prior frame or earlier in the current frame. The entity shall be positioned and oriented relative to the parent's coordinate system according to the *Yaw*, *Pitch*, *Roll*, *X Offset*, *Y Offset*, and *Z Offset* parameters (see Section 5.4.2). This parameter may be changed for a given entity at any time. The attachment or detachment shall take place immediately and remain in effect until changed with another **Entity Position** packet.
Ground/Ocean Clamp Type: unsigned 2-bit field Units: N/A Values: 0 No Clamp 1 Non-Conformal 2 Conformal	This parameter specifies whether the entity may be clamped to the ground or water surface. If the *Attach State* parameter of an **Entity Position** packet corresponding to the same *Entity ID* is set to Attach (1), the IG shall ignore this parameter. **No Clamp** – The entity is not clamped. The *Altitude* parameter shall specify the entity's height above Mean Sea Level. The *Pitch* and *Roll* parameters shall specify the entity's pitch and roll relative to the geodetic reference plane (Figure 18, page 46). **Non-Conformal** – The entity is clamped. The *Altitude* parameter shall specify an offset above the ground or water surface. The *Pitch* and *Roll* parameters shall specify the entity's pitch and roll relative to the geodetic reference plane (Figure 18, page 46). **Conformal** – The entity is clamped and its attitude shall conform to the terrain. The *Altitude* parameter shall specify an offset above the ground or water surface. The *Pitch* and *Roll* parameters shall specify the entity's pitch and roll relative to the slope of the terrain or water.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter identifies a specific entity. The IG shall apply the values in this packet to the state of the entity corresponding to this value.
Parent ID Type: unsigned int16 Units: N/A	This parameter specifies the parent entity to which this entity shall be attached. If the *Attach State* parameter is set to Detach (0), then the IG shall ignore this parameter. The value of this parameter may be changed without first detaching the entity from its existing parent. If the specified parent entity is invalid, then no change in the attachment shall be made.
Roll Type: single float Units: degrees Values: -180.0 – 180.0 Datum: If *Attach State* = 0 and *Ground/Ocean Clamp* = 0 or 1: Geodetic reference coordinate system (see Section 5.4.1.2) If *Attach State* = 0 and *Ground/Ocean Clamp* = 2: Surface polygon orientation If *Attach State* = 1: Entity reference coordinate system (see Section 5.4.2.2)	This parameter specifies the roll angle of the entity. For top-level entities for which the preceding *Ground/Ocean Clamp* parameter of an **Entity Control** packet for the same Entity ID was set to No Clamp (0) or Non-Conformal (1), this angle shall be measured from the reference plane described in Section 5.4.1.2. For top-level entities for which the preceding *Ground/Ocean Clamp* parameter of an **Entity Control** packet for the same Entity ID was enabled, this angle shall be measured from the ground or water surface polygon's orientation. For child entities, roll shall be measured from the entity's reference plane after yaw and pitch rotations have been applied as described in Section 5.4.2.2.

Parameter	Description
Pitch Type: single float Units: degrees Values: -90.0 – 90.0 Datum: If *Attach State* = 0 and *Ground/Ocean Clamp* = 0 or 1: Geodetic reference plane (see Section 5.4.1.2) If *Attach State* = 0 and *Ground/Ocean Clamp* = 2: Surface polygon orientation If *Attach State* = 1: Entity reference plane (see Section 5.4.2.2)	This parameter specifies the pitch angle of the entity. For top-level entities for which the preceding *Ground/Ocean Clamp* parameter of an **Entity Control** packet for the same Entity ID was set to No Clamp (0) or Non-Conformal (1), this angle shall be measured from the reference plane described in Section 5.4.1.2. For top-level entities for which the preceding *Ground/Ocean Clamp* parameter of an **Entity Control** packet for the same Entity ID was enabled, this angle shall be measured from the ground or water surface polygon's orientation. For child entities, pitch shall be measured with respect to the entity's reference plane after the yaw rotation has been applied as described in Section 5.4.2.2.
Yaw Type: single float Units: degrees Values: 0.0 – 360.0 Datum: If *Attach State* = 0: True North If *Attach State* = 1: Entity Reference Plane's +**X** axis	For top-level (non-child) entities, this parameter specifies the instantaneous heading of the entity. This angle shall be measured from a line parallel to the Prime Meridian as described in Section 5.4.1.2. For child entities, this parameter specifies the entity's rotation about its **Z** axis relative to the parent entity. This angle shall be measured from a line passing through the child's reference point and that is parallel to the parent's **X** axis as described in Section 5.4.2.2.
Latitude (Top-Level Entities) Type: double float Units: degrees Values: -90 – 90 Datum: Equator	For top-level (non-child) entities, this parameter specifies the entity's geodetic latitude. This angle shall be measured from the equatorial plane to an imaginary vector that is normal to the ellipsoid and passes through the entity's reference point as described in Section 5.4.1.1.
X Offset (Child Entities) Type: double float Units: meters Datum: Parent's reference point	For child entities, this parameter specifies the child's X offset from the parent entity. This offset shall be measured from the parent's reference point along the parent's **X** axis.

Parameter	Description
Longitude (Top-Level Entities) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	For top-level (non-child) entities, this parameter specifies the entity's geodetic longitude. This angle shall be measured from the Prime Meridian to an imaginary vector that is normal to the ellipsoid and passes through the entity's reference point as described in Section 5.4.1.1.
Y Offset (Child Entities) Type: double float Units: meters Datum: Parent's reference point	For child entities, this parameter specifies the child's Y offset from the parent entity. This offset shall be measured from the parent's reference point along the parent's **Y** axis.
Altitude (Top-Level Entities) Type: double float Units: meters Datum: If *Ground/Ocean Clamp* = 0: Mean Sea Level If *Ground/Ocean Clamp* = 1 or 2: Ground/sea surface elevation	For top-level (non-child) entities, this parameter specifies the entity's altitude. If the *Ground/Ocean Clamp* parameter is set to No Clamp (0), this distance shall be measured above Mean Sea Level. If the *Ground/Ocean Clamp* parameter is set to Non-Conformal (1) or Conformal (2), this distance shall be measured above the ground or water surface (see Section 6.1.13) at the entity's latitude and longitude.
Z Offset (Child Entities) Type: double float Units: meters Datum: Parent's reference point	For child entities, this parameter specifies the child's Z offset from the parent entity. This offset shall be measured from the parent's reference point along the parent's **Z** axis.

6.1.3 Conformal Clamped Entity Position

The **Conformal Clamped Entity Position** packet may be used to reposition conformal ground- or ocean-clamped entities. This packet is offered as a lightweight alternative to the **Entity Position** packet (Section 6.1.2).

Before using this packet to manipulate an entity, the Host shall first instantiate that entity by sending an **Entity Control** packet with the *Ground/Ocean Clamp* parameter set to Conformal (2). If a non-existent entity is referenced by a **Conformal Clamped Entity Position** packet, the IG shall ignore the packet.

An entity's current roll, pitch, and altitude offsets shall be maintained when the IG receives a **Conformal Clamped Entity Position** packet describing that entity. If this packet is applied to an unclamped or non-conformal clamped entity, its current absolute roll, pitch, and altitude shall be maintained.

The contents of the **Conformal Clamped Entity Position** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 02h															
Entity ID																Reserved															
Yaw																															
Reserved																															
Latitude																															
Longitude																															

Figure 42 – Conformal Clamped Entity Position Packet Structure

Table 7 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 7 – Conformal Clamped Entity Position Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 02h	This parameter identifies this data packet as the **Conformal Clamped Entity Position** packet. The Host shall set this parameter to 02h.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the packet data shall be applied.
Yaw Type: single float Units: degrees Values: 0.0 – 360.0 Datum: True North	This parameter specifies the instantaneous heading of the entity. This angle shall be measured from True North as described in Section 5.4.1.2.
Latitude Type: double float Units: degrees Values: -90 – 90 Datum: Equator	This parameter shall specify the entity's geodetic latitude. This angle shall be measured from the equatorial plane to an imaginary vector that is normal to the ellipsoid and passes through the entity's reference point as described in Section 5.4.1.1.
Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	This parameter specifies the entity's geodetic longitude. This angle shall be measured from the Prime Meridian to an imaginary vector that is normal to the ellipsoid and passes through the entity's reference point as described in Section 5.4.1.1.

6.1.4 Component Control

The **Component Control** packet is provided as a generic mechanism to control various components within the simulation. Because CIGI is designed to be a general-purpose interface, only the most common attributes of entities, views, and other objects are explicitly represented by parameters within specific data packets. Other attributes should be represented by components.

Components may correspond to parts or properties of entities, views and view groups, sensors, weather and environment, terrain, and even the IG itself. The type of object possessing the component is identified by the *Component Class* parameter. The specific instance of that object is specified by the *Instance ID* parameter. Depending upon the value of the *Component Class* parameter, the instance ID might correspond to an Entity ID, a view ID, an environmental attribute, etc. The *Component ID* parameter uniquely identifies the component for the instance.

Table 8 lists each component class, the identifier to which *Instance ID* corresponds for that class, and examples of component ID assignments with possible values. Note that the table gives only one example of a possible component assignment scheme.

Table 8 – Examples of Component Assignments

Component Class	*Instance ID* Corresponds to	Example *Component ID* Assignments	Example States and Values
Entity	Entity ID	Anti-collision light	On/Off
		Air brake	Full/Partial/Closed
		Landing gear	Up/Down
		Damage	Damaged/No Damage
		Entity Scaling	Enable/Disable, X, Y, Z
		Engine Temperature	Temperature differential
View	View ID	Zoom	Zoom factor
		Viewport Mask	On/Off
		Angle of Projection (for an oblique parallel view)	Angle
View Group	Group ID	Zoom	Zoom Factor
Sensor	Sensor ID	Gate Cursor	Symbol
Regional Sea Surface	Region ID	Littoral Current	Speed, Direction
Regional Terrain Surface	Region ID	Light Snow Properties	Wetness
		Heavy Snow Properties	Depth, Wetness
		Mud Properties	Depth, Consistency, Stability
Regional Layered Weather	Region ID	Instantaneous Lightning Strike (Layer 1)	Latitude, Longitude, Altitude, Intensity
		Instantaneous Lightning Strike (Layer 2)	Latitude, Longitude, Altitude, Intensity
		Rain (Layer 4)	Rain Density, Slant
		Aerosol (Layer 10)	Color
		Aerosol (Layer 11)	Color

Component Class	*Instance ID* Corresponds to	Example *Component ID* Assignments	Example States and Values
Global Sea Surface	–	Surf Zone Breakers	Breaker Height, Direction, Period
		Littoral Current	Speed, Direction
Global Terrain Surface	–	Runway Lights	On/Off, Intensity, Punch-Through
		Cultural Lights Group 1	On/Off, Intensity, Color
		Cultural Lights Group 2	On/Off, Intensity, Color
		Shadows	On/Off, Contrast
Global Layered Weather	Layer ID	Instantaneous Lightning Strike	Latitude, Longitude, Altitude, Intensity
		Random Lightning	Enable/Disable, Avg. Freq.
		Aerosol	Color, Density, Reflectivity
Atmosphere	–	Haze	On/Off, Haze Color
Celestial Sphere	–	Sky Color	Color
Event	Event ID	Event Properties	Enable/Disable, Period
System	–	Crash Indicator	On/Off, Color
		High-Resolution Targets	Enable/Disable
Symbol Surface	Symbol Surface ID	Surface Background Color	RGBA Color
Symbol	Symbol ID	Line End-Cap Style	Round, Square
		Texture	Enable/Disable, Texture ID

Each component may have one or more discrete states and up to six user-defined values. The user-defined values may be integers, real numbers, or multiple fields subdivided according to the rules below. A light, for instance, might have two discrete states (On and Off), an RGB color value represented as a 32-bit integer, and a real-value intensity level. A component representing the global lightpoint intensity value might have a real-value intensity level and no discrete states.

Regardless of how the user-defined data fields are formatted, they shall be byte-swapped as separate 32-bit values if the Host and IG use different byte ordering schemes. This ensures that all **Component Control** packets are byte-swapped in a consistent manner. The data should be packaged using masks and bit-wise operations so that the values are not changed when the 32-bit words are byte-swapped.

As an example, assume the *Component Data 1* field is used to store two 16-bit integers, i and j; the *Component Data 2* field is used to store a single 32-bit integer, k; and the *Component Data 3* and *Component Data 4* fields are combined to store a double-precision floating-point number, m. The two remaining parameters are not used in this instance. Figure 43 illustrates an incorrect and a correct way of formatting the 32-bit user-defined data fields to store these values:

Incorrect:

Instance ID	Reserved
Reserved	
i	***j***
k	
m	
0	
0	

Correct:

Instance ID	Reserved
Reserved	
$(i \times 2^{16}) + j$	
k	
***m* (MSW)**	
***m* (LSW)**	
0	
0	

Figure 43 – Example of Incorrect and Correct Formatting of Component Data

Since k is a 32-bit data type, its value can simply be copied to the packet. However, if the Host were to copy the values of i, j, and m as shown in the top structure, these data would be improperly byte-swapped by the IG. For this example, assume that the Host uses little-endian byte ordering and that the IG is a big-endian system. If i were 1, j were 120, k were 3600, and m were 100.0, the data would be incorrectly interpreted by the IG as shown in the following diagram:

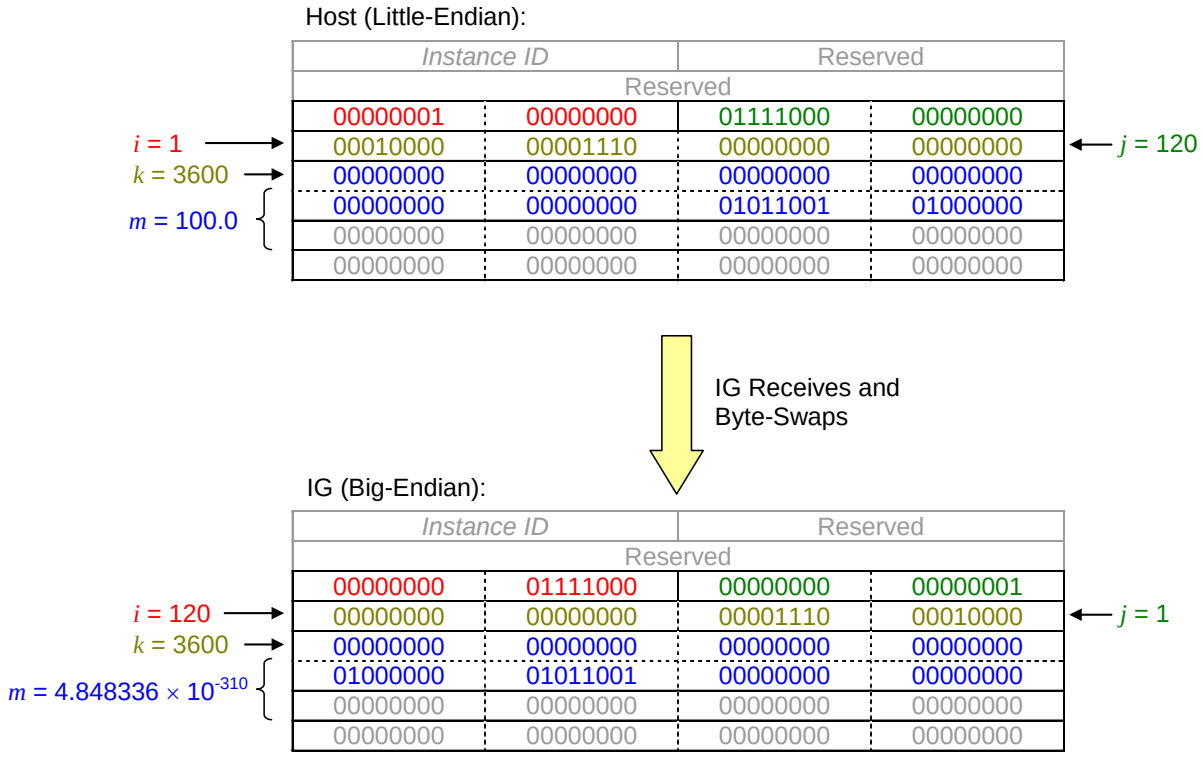


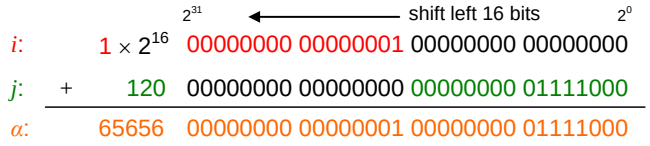
Figure 44 – Example of Incorrect Packaging of Component Data

In this scenario, the values of i and j have been swapped. In addition, the value of the floating-point number m has changed because the most and least significant words have also been swapped.

When the Host packs the data into the **Component Control** packet, it should combine the 16-bit integers into a single 32-bit number, α, by bit-shifting i left 16 bits (or multiplying i by 2^{16}) and adding j. It should then split the 64-bit floating-point number into two 32-bit values, β and γ, which represent the most significant word (MSW) and least significant word (LSW), respectively. The value of β can be found by truncating m after the 32nd bit (2^{31}), and the value of γ by shifting m to the right 32 bits and truncating after the 32nd bit.

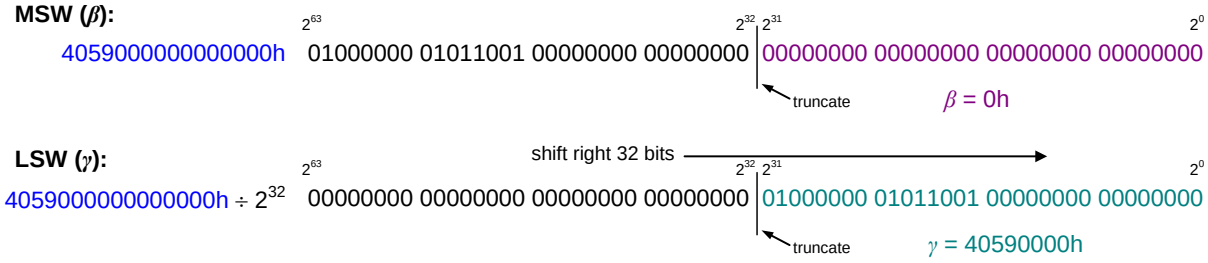
The diagram below illustrates the packing of the data into the correct format shown in Figure 43. Note that to find the least-significant word of m, the variable is typecast as an unsigned 64-bit integer.

Pack the 16-bit Integers:



Pack the 64-bit Floating-Point Number:

m = 100.0 can be represented as 4059000000000000h, which is treated as an unsigned 64-bit integer



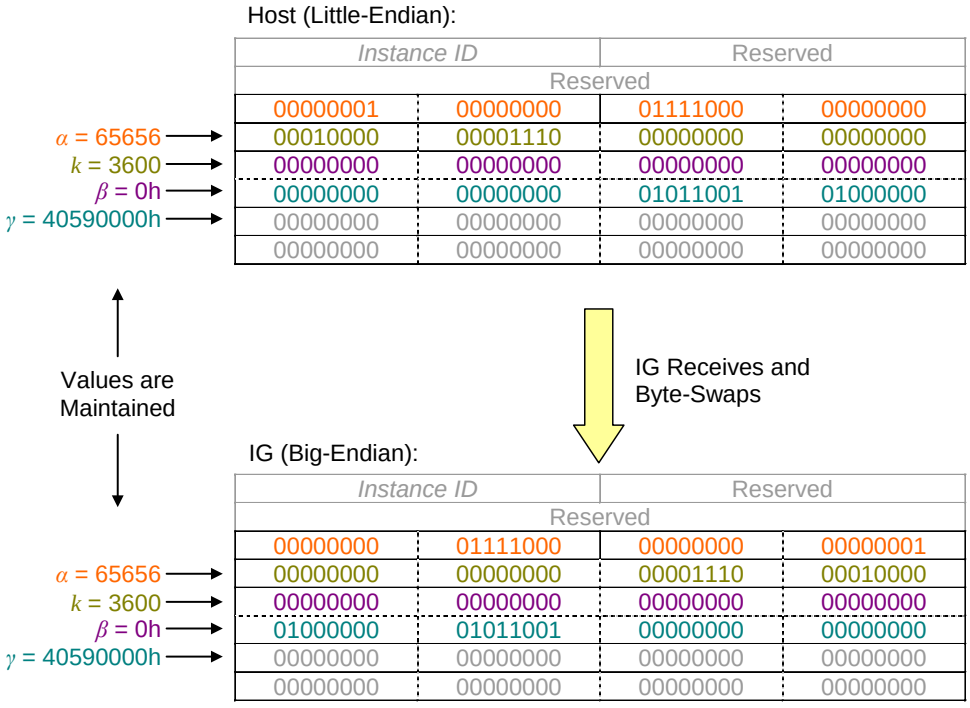


Figure 45 – Example of Correct Packaging of Component Data

k is a 32-bit data type and shall be byte-swapped correctly without manipulation. Since α, β, and γ are 32-bit fields, their values shall also be maintained when the IG byte-swaps the **Component Control** packet. The IG can correctly reconstruct the values of i, j, and k by applying the reciprocal operations to α, β, and γ after byte-swapping.

Single-byte int8 data and UTF-8 code points would be packed in a method similar to that used for *i* and *j*. Each numerical value would be shifted left (multiplied by a power of two) and added to the unsigned 32-bit integer.

Because the method of organizing *n*-bit fields varies from one compiler to the next, *n*-bit fields should be implemented with a series of bit-wise shifts and logical operations.

This packet uses the same *Component ID* and *Instance ID* mappings as the **Short Component Control** packet (Section 6.1.5). All components that may be controlled with the **Short Component Control** packet may also be controlled with the **Component Control** packet.

The contents of the **Component Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 40																*Packet ID* = 03h															
Component ID								*1								*Component Class*								*Component State*							
Instance ID																Reserved															
Reserved																															
Component Data 1																															
Component Data 2																															
Component Data 3																															
Component Data 4																															
Component Data 5																															
Component Data 6																															

*1 Reserved

Figure 46 – Component Control Packet Structure

Table 9 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 9 – Component Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 40.
Packet ID Type: unsigned int16 Units: N/A Value: 03h	This parameter identifies this data packet as the **Component Control** packet. The Host shall set this parameter to 03h.

Parameter	Description
Component ID Type: unsigned int16 Units: N/A	The value of this parameter uniquely identifies the component to which the data in this packet shall be applied. If *Component Class* is set to Regional Layered Weather (6), the weather layer ID shall be specified by the most significant byte of *Component ID*.
Component Class Type: unsigned 6-bit field Units: N/A Values: 0 Entity 1 View 2 View Group 3 Sensor 4 Regional Sea Surface 5 Regional Terrain Surface 6 Regional Layered Weather 7 Global Sea Surface 8 Global Terrain Surface 9 Global Layered Weather 10 Atmosphere 11 Celestial Sphere 12 Event 13 System 14 Symbol Surface 15 Symbol 16–63 Reserved	This parameter specifies the type of object to which the *Instance ID* parameter refers. Both of these parameters shall be used in conjunction with *Component ID* to uniquely identify a component in the simulation (see Table 8).
Component State Type: unsigned int8 Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter specifies a discrete state for the component. For example, the value of this parameter may correspond to the state of a switch node within a model hierarchy. The IG shall activate the specified state for the component. If the state is invalid, then the IG shall ignore the packet.

Parameter	Description
Instance ID Type: unsigned int16 Units: N/A	The value of this parameter uniquely identifies the object to which the component belongs. This value shall correspond to an entity ID, a view or view group ID, a sensor ID, environmental region ID, global weather layer ID, event ID, or System ID depending upon the value of the *Component Class* parameter (see Table 8). This parameter shall be ignored if *Component Class* is set to Global Sea Surface (7), Global Terrain Surface (8), Atmosphere (10), or Celestial Sphere (11).
Component Data 1 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.
Component Data 2 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.
Component Data 3 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.

Parameter	Description
Component Data 4 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.
Component Data 5 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.
Component Data 6 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.

6.1.5 Short Component Control

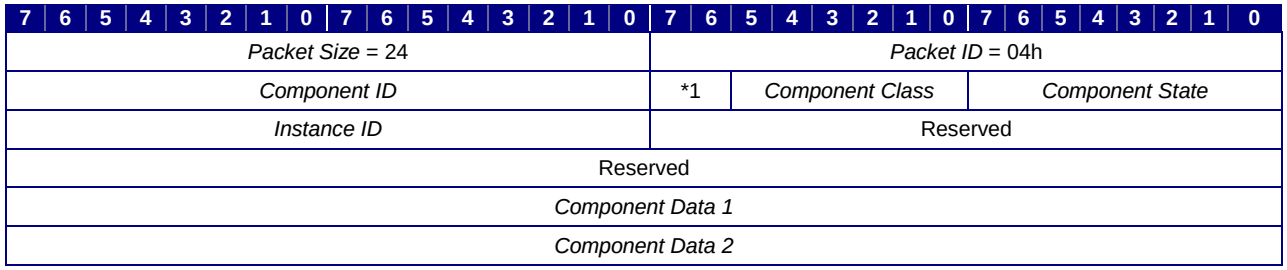
The **Short Component Control** packet, like the **Component Control** packet (Section 6.1.4), is a generic packet that may be used to control a variety of objects or functions on the IG. This packet is provided as a lower-bandwidth alternative to the **Component Control** packet for components that do not require more than two words of user-defined component data.

This packet uses the same *Component ID* and *Instance ID* mappings as the **Component Control** packet. If the additional data fields offered by the **Component Control** packet are not necessary for a component, then the two packet types shall be interchangeable. In other words, all components that may be controlled with the **Short Component Control** packet shall also be controllable with the **Component Control** packet.

When receiving a **Short Component Control** packet, the IG may copy the contents of the packet into a **Component Control** structure, padding the remainder of the packet with zeros (0), thereby allowing the two packet types to be processed by the same packet-handling routine.

The *Component Data 1* and *Component Data 2* fields shall be byte-swapped, if necessary, as 32-bit data types. Data shall be packed into 32-bit units as described in Section 6.1.4.

The contents of the **Short Component Control** packet are as follows:



*1 Reserved

Figure 47 – Short Component Control Packet Structure

Table 10 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 10 – Short Component Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 04h	This parameter identifies this data packet as the **Short Component Control** packet. The Host shall set this parameter to 04h.
Component ID Type: unsigned int16 Units: N/A	The value of this parameter uniquely identifies the component to which the data in this packet shall be applied. If *Component Class* is set to Regional Layered Weather (6), the weather layer ID shall be specified by the most significant byte of *Component ID*.
Component Class Type: unsigned 4-bit field Units: N/A Values: 0 Entity 1 View 2 View Group 3 Sensor 4 Regional Sea Surface 5 Regional Terrain Surface 6 Regional Layered Weather 7 Global Sea Surface 8 Global Terrain Surface 9 Global Layered Weather 10 Atmosphere 11 Celestial Sphere 12 Event 13 System 14 Symbol Surface 15 Symbol 16–63 Reserved	This parameter identifies the type of object to which the *Instance ID* parameter refers. Both of these parameters are used in conjunction with *Component ID* to uniquely identify a component in the simulation (see Table 8).
Component State Type: unsigned int8 Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter specifies a discrete state for the component. For example, the value of this parameter may correspond to the state of a switch node within a model hierarchy. The IG shall activate the specified state for the component. If the state is invalid, then the IG shall ignore the parameter.

Parameter	Description
Instance ID Type: unsigned int16 Units: N/A	The value of this parameter uniquely identifies the object to which the component belongs. This value corresponds to an entity ID, a view or view group ID, a sensor ID, environmental region ID, global weather layer ID, or event ID depending upon the value of the *Component Class* parameter (see Table 8). This parameter may typically be ignored if *Component Class* is set to Global Sea Surface (7), Global Terrain Surface (8), Atmosphere (10), Celestial Sphere (11), or System (13).
Component Data 1 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.
Component Data 2 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of six 32-bit words used for user-defined component data. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple data, then those data shall be packed without regard to physical layout so the IG is able to extract the values regardless of whether byte swapping has occurred.

6.1.6 Articulated Part Control

Articulated parts are entity features that may be rotated and/or translated with respect to the entity. These features are submodels of the entity model and possess their own coordinate systems as discussed in Section 5.4.3. Examples include wing flaps, landing gear, and tank turrets.

Articulated parts may be manipulated in up to six degrees of freedom. Translation shall be defined as X, Y, and Z offsets relative to the submodel's reference point. Rotation shall be defined relative to the submodel coordinate system.

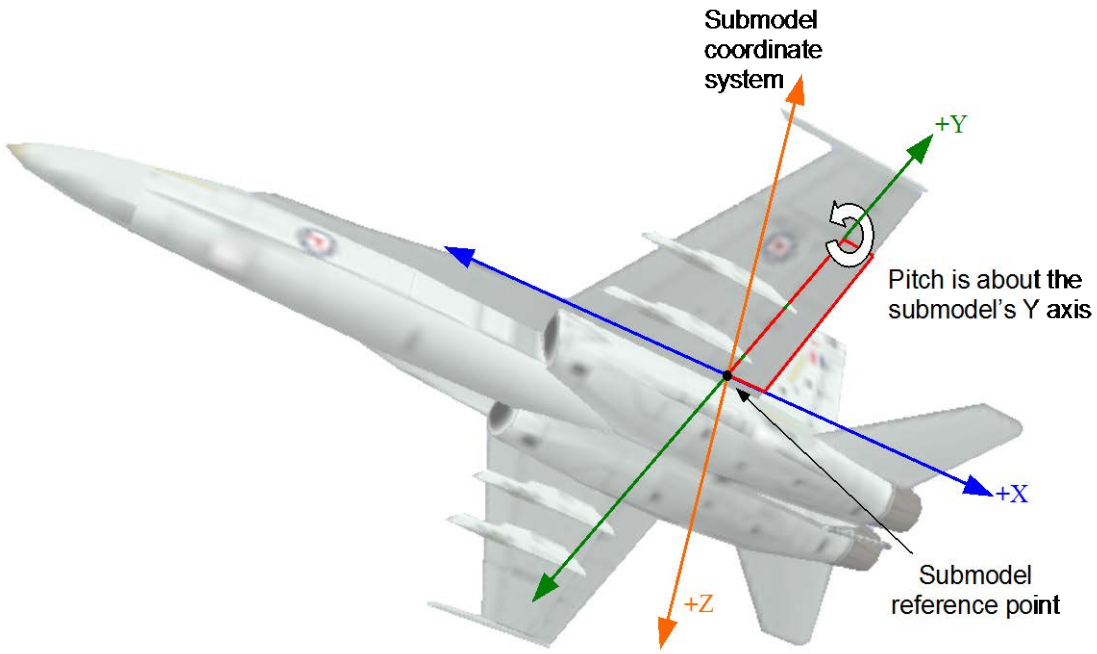


Figure 48 – Manipulation of Articulated Part Submodel

Positional and rotational values shall not be cumulative. They are absolute values relative to the coordinate system defined within the model.

The contents of the **Articulated Part Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 05h							
EntityID																*Articulated Part ID*							
X Offset																							
Y Offset																							
Z Offset																							
Roll																							
Pitch																							
Yaw																							

*1	*7	*6	*5	*4	*3	*2	*1
*8								

*1 *Articulated Part Enable*
*2 *X Offset Enable*
*3 *Y Offset Enable*
*4 *Z Offset Enable*
*5 *Roll Enable*
*6 *Pitch Enable*
*7 *Yaw Enable*
*8 Reserved

Figure 49 – Articulated Part Control Packet Structure

Table 11 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 11 – Articulated Part Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 05h	This parameter identifies this data packet as the **Articulated Part Control** packet. The Host shall set this parameter to 05h.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity whose articulated part submodel, specified by *Articulated Part ID*, shall be manipulated.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter specifies the articulated part submodel to which the data in this packet shall be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the model hierarchy.

Parameter	Description
Articulated Part Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter determines whether the articulated part submodel will be enabled or disabled. If this parameter is set to Disable (0), the part shall be hidden within the scene. If the parameter is set to Enable (1), the part shall be included in the scene.
X Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *X Offset* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), then the *X Offset* parameter shall be ignored and the articulated part shall remain at its current location along the submodel's **X** axis. If this parameter is set to Enable (1), then the articulated part submodel shall be positioned along the submodel's **X** axis to the position specified by the *X Offset* parameter.
Y Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Y Offset* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), *Y Offset* shall be ignored and the articulated part shall remain at its current location along the submodel's **Y** axis. If this parameter is set to Enable (1), then the articulated part submodel shall be positioned along the submodel's **Y** axis to the position specified by the *Y Offset* parameter.
Z Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Z Offset* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), *Z Offset* shall be ignored and the articulated part shall remain at its current location along the submodel's **Z** axis. If this parameter is set to Enable (1), then the articulated part submodel shall be positioned along the submodel's **Z** axis to the position specified by the *Z Offset* parameter.

Parameter	Description
Roll Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Roll* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), *Roll* shall be ignored and the articulated part shall retain its current roll angle. If this parameter is set to Enable (1), then the articulated part submodel shall be rotated about the submodel's **X** axis to the angle specified by the *Roll* parameter.
Pitch Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Pitch* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), *Pitch* shall be ignored and the articulated part shall retain its current pitch angle. If this parameter is set to Enable (1), then the articulated part submodel shall be rotated along the submodel's **Y** axis to the angle specified by the *Pitch* parameter.
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Yaw* parameter of the current packet may be applied to the articulated part. If this parameter is set to Disable (0), *Yaw* shall be ignored and the articulated part shall retain its current yaw angle. If this parameter is set to Enable (1), then the articulated part submodel shall be rotated along the submodel's **Z** axis to the angle specified by the *Yaw* parameter.
X Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the X offset of the articulated part submodel. This position shall be measured from the submodel's reference point along its **X** axis. If *X Offset Enable* is set to Disable (0), then *X Offset* shall be ignored and the articulated part shall retain its current X offset or rate.

Parameter	Description
Y Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the Y offset of the articulated part submodel. This position shall be measured from the submodel's reference point along its **Y** axis. If *Y Offset Enable* is set to Disable (0), then *Y Offset* shall be ignored and the articulated part shall retain its current Y offset or rate.
Z Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the Z offset of the articulated part submodel. This position shall be measured from the submodel's reference point along its **Z** axis. If *Z Offset Enable* is set to Disable (0), then *Z Offset* shall be ignored and the articulated part shall retain its current Z offset or rate.
Roll Type: single float Units: degrees Values: -180.0 – 180.0 Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **X** axis. This angle shall be measured from the submodel's **XY** plane after yaw and pitch have been applied. If *Roll Enable* is set to Disable (0), then *Roll* shall be ignored and the articulated part shall retain its current roll angle or rate.
Pitch Type: single float Units: degrees Values: -180.0 – 180.0 Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **Y** axis. This angle shall be measured from the submodel's **XY** plane after yaw has been applied. If *Pitch Enable* is set to Disable (0), then *Pitch* shall be ignored and the articulated part shall retain its current pitch angle or rate.

Parameter	Description
Yaw Type: single float Units: degrees Values: -180.0 – 180.0 Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **Z** axis. This angle shall be measured from the submodel's +**X** axis before yaw and pitch have been applied. If *Yaw Enable* is set to Disable (0), then *Yaw* shall be ignored and the articulated part shall retain its current yaw angle or rate.

6.1.7 Short Articulated Part Control

The **Short Articulated Part Control** packet is provided as a lower-bandwidth alternative to the **Articulated Part Control** packet (Section 6.1.6). It may be used when manipulation of only one or two degrees of freedom of a submodel are necessary.

This packet allows for up to two articulations. The articulations may be applied to a single articulated part or two separate ones belonging to the same entity. The articulated part or parts are specified by the *Articulated Part ID 1* and *Articulated Part ID 2* parameters. Two floating-point degree-of-freedom (DOF) parameters, *DOF 1* and *DOF 2*, specify offsets or angular positions for the specified articulated parts. The *DOF Select 1* and *DOF Select 2* parameters specify which degree of freedom each of these floating-point parameters represents.

Note: If *DOF Select 1* and *DOF Select 2* refer to the same degree of freedom for the same articulated part, then *DOF 2* (i.e.,, the “last-in” value) shall take priority over *DOF 1*.

The contents of the **Short Articulated Part Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 24</td><td colspan="16">Packet ID = 06h</td></tr>
<tr><td colspan="16">Entity ID</td><td colspan="8">Articulated Part ID 1</td><td colspan="8">Articulated Part ID 2</td></tr>
<tr><td>*4</td><td>*3</td><td colspan="2">*2</td><td colspan="2">*1</td><td colspan="26">Reserved</td></tr>
<tr><td colspan="32">DOF 1</td></tr>
<tr><td colspan="32">DOF 2</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *DOF Select 1*
*2 *DOF Select 2*
*3 *Articulated Part Enable 1*
*4 *Articulated Part Enable 2*

Figure 50 – Short Articulated Part Control Packet Structure

Table 12 defines each parameter’s data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 12 – Short Articulated Part Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 06h	This parameter identifies this data packet as the **Short Articulated Part Control** packet. The Host shall set this parameter to 06h.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity whose articulated part submodel(s) shall be manipulated.
Articulated Part ID 1 Type: unsigned int8 Units: N/A	This parameter specifies one of up to two articulated parts to which the data in this packet shall be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the model hierarchy. The value of this parameter may be equal to that of *Articulated Part ID 2*.
Articulated Part ID 2 Type: unsigned int8 Units: N/A	This parameter specifies one of up to two articulated parts to which the data in this packet shall be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the model hierarchy. The value of this parameter may be equal to that of *Articulated Part ID 1*.
DOF Select 1 Type: 3-bit field Units: N/A Values: 0 Not Used 1 X Offset 2 Y Offset 3 Z Offset 4 Yaw 5 Pitch 6 Roll	This parameter specifies the degree of freedom to which the value of *DOF 1* shall be applied. If this parameter is set to Not Used (0), *DOF 1* shall be ignored.
DOF Select 2 Type: 3-bit field Units: N/A Values: 0 Not Used 1 X Offset 2 Y Offset 3 Z Offset 4 Yaw 5 Pitch 6 Roll	This parameter specifies the degree of freedom to which the value of *DOF 2* shall be applied. If this parameter is set to Not Used (0), *DOF 2* shall be ignored.

Parameter	Description
Articulated Part Enable 1 Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter determines whether the articulated part submodel specified by *Articulated Part ID 1* may be enabled or disabled. If this parameter is set to Disable (0), the part shall be removed from the scene. If the parameter is set to Enable (1), the part shall be included in the scene.
Articulated Part Enable 2 Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter determines whether the articulated part submodel specified by *Articulated Part ID 2* may be enabled or disabled within the model hierarchy. If this parameter is set to Disable (0), the part shall be removed from the scene. If the parameter is set to Enable (1), the part shall be included in the scene.
DOF 1 Type: single float Units: meters or degrees (see description at right) Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies either an offset or an angular position for the part identified by *Articulated Part ID 1*. The interpretation of this value is determined by the *DOF Select 1* parameter. If the parameter is set to X Offset (1), Y Offset (2), or Z Offset (3), then *DOF 1* shall specify an offset in meters. If *DOF Select 1* is set to Yaw (4), Pitch (5), or Roll (6), then *DOF 1* shall specify an angular position in -180 to 180 degrees.
DOF 2 Type: single float Units: meters or degrees (see description at right) Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies either an offset or an angular position for the part identified by *Articulated Part ID 2*. The interpretation of this value is determined by the *DOF Select 2* parameter. If the parameter is set to X Offset (1), Y Offset (2), or Z Offset (3), then *DOF 2* shall specify an offset in meters. If *DOF Select 2* is set to Yaw (4), Pitch (5), or Roll (6), then *DOF 2* shall specify an angular position in -180 to 180 degrees.

6.1.8 Velocity Control

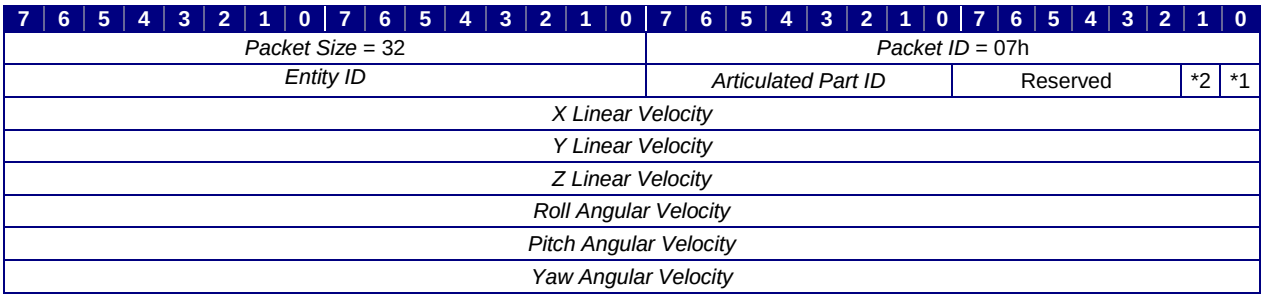
The **Velocity Control** packet is used to define linear and angular velocities for entities and articulated parts.

The **Velocity Control** packet is useful for models and submodels whose behavior is predictable and whose exact positions need not be known each frame by the Host. A rotating radar dish on a ground target, for example, revolves in a consistent manner and the Host typically does not need to know its instantaneous yaw angle.

Velocities may also be used to enable the IG to compensate for transport delays or jitter produced by asynchronous operation. A **Velocity Control** packet may be sent each frame in conjunction with an **Entity Position** (6.1.2) packet or alone for an active entity that has been positioned previously with an **Entity Position** packet. This provides the IG with velocity information to extrapolate the entity's probable position during the next frame if smoothing has been enabled for the entity in the **Entity Control** packet (6.1.38).

When a velocity is specified for an entity or articulated part, the IG shall maintain that velocity until a new velocity is specified by the Host. If the Host changes the position and/or orientation of an entity or articulated part, the IG shall perform the transformation and extrapolation shall continue from that state beginning with the next frame. If the Host sets all velocity components to zero, the entity or articulated part shall no longer be extrapolated.

The contents of the **Velocity Control** packet are as follows:



*1 *Apply to Articulated Part*
*2 *Coordinate System*

Figure 51 – Velocity Control Packet Structure

Table 13 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 13 – Velocity Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 07h	This parameter identifies this data packet as the **Velocity Control** packet. The Host shall set this parameter to 07h.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the velocity may be applied. If the Host sets the *Apply to Articulated Part* flag to True (1), then the IG shall apply the velocity to an articulated part belonging to this entity. If the Host sets the flag to False (0), then the IG shall apply the velocity to the whole entity.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter specifies the articulated part to which the velocity may be applied. If the *Apply to Articulated Part* flag is set to True (1), this parameter shall identify the articulated part belonging to the entity specified by *Entity ID*. If the flag is set to False (0), the IG shall ignore this parameter.
Apply to Articulated Part Type: 1-bit field Units: N/A Values: 0 False 1 True	This parameter specifies whether the velocity may be applied to an articulated part or an entity. If this flag is set to True (1), the velocity shall be applied to the articulated part specified by the *Articulated Part ID*. If this flag is set to False (0), the velocity shall be applied to the entity specified by the *Entity ID* parameter.

Parameter	Description
Coordinate System Type: 1-bit field Units: N/A Values: 0 World/Parent 1 Local	This parameter specifies the reference coordinate system to which the linear and angular velocities may be applied. When this parameter is set to World/Parent (0) and the entity is a top-level (non-child) entity, the velocities shall be defined relative to the Local Geodetic Reference Plane with NED Coordinate System as described in Section 5.4.1.2. Linear velocities shall describe a path along and above the surface of the geoid and angular velocities shall describe a rotation relative to the axes of the reference plane as shown in Figure 18. When this parameter is set to World/Parent (0) and the entity is a child entity, the velocities shall be defined relative to the parent's local coordinate system as described in Section 5.4.2.2. When this parameter is set to Local (1), the velocities shall be defined relative to the entity's local coordinate system. This parameter shall be ignored if *Apply to Articulated Part* is set to True (1).
X Linear Velocity Type: single float Units: m/s Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the X component of a linear velocity vector. The IG shall continually move the specified entity or articulated part along this linear velocity vector.
Y Linear Velocity Type: single float Units: m/s Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the Y component of a linear velocity vector. The IG shall continually move the specified entity or articulated part along this linear velocity vector.

Parameter	Description
Z Linear Velocity Type: single float Units: m/s Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the Z component of a linear velocity vector. The IG shall continually move the specified entity or articulated part along this linear velocity vector.
Roll Angular Velocity Type: single float Units: deg/s Default: Model-dependent Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **X** axis after yaw and pitch have been applied. The IG shall rotate the specified entity or articulated part at this angular velocity and shall honor the order of rotation described in Section 5.4.
Pitch Angular Velocity Type: single float Units: deg/s Default: Model-dependent Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **Y** axis after yaw has been applied. The IG shall rotate the specified entity or articulated part at this angular velocity and shall honor the order of rotation described in Section 5.4.
Yaw Angular Velocity Type: single float Units: deg/s Default: Model-dependent Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the angle of rotation of the articulated part about its **Z** axis when its **X** axis is parallel to that of the entity. The IG shall rotate the specified entity or articulated part at this angular velocity and shall honor the order of rotation described in Section 5.4.

6.1.9 Celestial Sphere Control

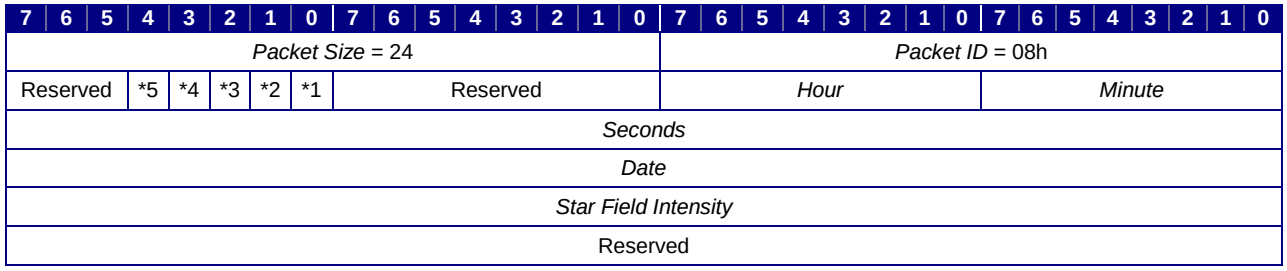
The **Celestial Sphere Control** data packet allows the Host to specify properties of the sky model.

The *Date* parameter specifies the current date and the *Hour* and *Minute* and *Second*s parameters specify the current time of day. The IG uses these parameters to determine ambient light properties, sun and moon positions (and corresponding directional light positions), moon phase, and horizon glow.

An IG typically uses an ephemeris model to continuously update the time of day. A **Celestial Sphere Control** packet need not be sent each minute for the sole purpose of updating the time of day unless the Host has disabled the ephemeris model with the *Continuous Time-of-day Enable* flag.

Note: If the Host intends to freeze the entire simulation, it should send a **Celestial Sphere Control** packet with the *Continuous Time-of-day Enable* parameter set to Disable (0); otherwise, the IG shall continue to update the time of day. When the Host resumes the simulation, it should explicitly re-enable the ephemeris model.

The contents of the **Celestial Sphere Control** packet are as follows:



*1 *Continuous Time-of-day Enable*
*2 *Sun Enable*
*3 *Moon Enable*
*4 *Star Field Enable*
*5 *Date/Time Valid*

Figure 52 – Celestial Sphere Control Packet Structure

Table 14 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 14 – Celestial Sphere Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 08h	This parameter identifies this data packet as the **Celestial Sphere Control** packet. The Host shall set this parameter to 08h.
Continuous Time-of-Day Enable Type: 1-bit field Units: N/A Values: 0 Disable (Static time of day) 1 Enable (Continuous time of day) Default: Enable	The value of this parameter controls whether the time of day is static or continuous. If this parameter is set to Disabled (0), the image generator shall not continuously update the time of day. If this parameter is set to Enabled (1), the image generator shall continuously update the time of day.
Sun Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter, along with the appropriate time-of-day and location on the globe specifies whether the sun is seen in the sky. If this parameter is set to Disable (0), the sun shall not be seen in the sky. If this parameter is set to Enable (1), the sun shall be seen in the sky.
Moon Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter, along with the appropriate time-of-day and location on the globe specifies whether the moon is seen in the sky. If this parameter is set to Disable (0), the moon shall not be seen in the sky. If this parameter is set to Enable (1), the moon shall be seen in the sky. The moon phase shall be determined by the current date.
Star Field Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter, along with the appropriate time-of-day and location on the globe specifies whether stars are seen in the sky. If this parameter is set to Disable (0), the stars shall not be seen in the sky. If this parameter is set to Enable (1), the stars shall be seen in the sky. The star positions shall be determined by the current date, time-of-day and location on the globe.

Parameter	Description
Date/Time Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	The value of this parameter may determine whether the *Hour*, *Minute*, and *Date* parameters are valid. If *Date/Time Valid* is set to Invalid (0), these values shall be ignored. If *Date/Time Valid* is set to Valid (1), these values shall override the IG's current date and time.
Hour Type: unsigned int8 Units: hours Values: 0 – 23 Default: 0 Datum: UTC	This parameter specifies the current hour of the day within the simulation. If the *Date/Time Valid* parameter is set to Valid (1), then the IG shall set the date and time as specified in this packet. If the *Date/Time Valid* parameter is set to Invalid (0), then the IG shall ignore this parameter.
Minute Type: unsigned int8 Units: minutes Values: 0 – 59 Default: 0 Datum: UTC	This parameter specifies the current minute of the day within the simulation. If the *Date/Time Valid* parameter is set to Valid (1), then the IG shall set the date and time as specified in this packet. If the *Date/Time Valid* parameter is set to Invalid (0), then the IG shall ignore this parameter.
Seconds Type: single float Units: Seconds Values: 0.0 – 59.999999 Default: 0 Datum: UTC	This parameter specifies the current seconds of the minute within the simulation. If the *Date/Time Valid* parameter is set to Valid (1), then the IG shall set the date and time as specified in this packet. If the *Date/Time Valid* parameter is set to Invalid (0), then the IG shall ignore this parameter.
Date Type: unsigned int32 Units: N/A Values: See description at right Default: IG-configurable	The value of this parameter specifies the current date within the simulation. The date shall be represented as a seven- or eight-digit decimal integer formatted as follows: YYYYMMDD = (year × 10000) + (month × 100) + day

Parameter	Description
Star Field Intensity Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall be applied as a percentage of the maximum intensity of each star in the star field. This parameter shall be ignored if *Star Field Enable* is set to Disable (0).

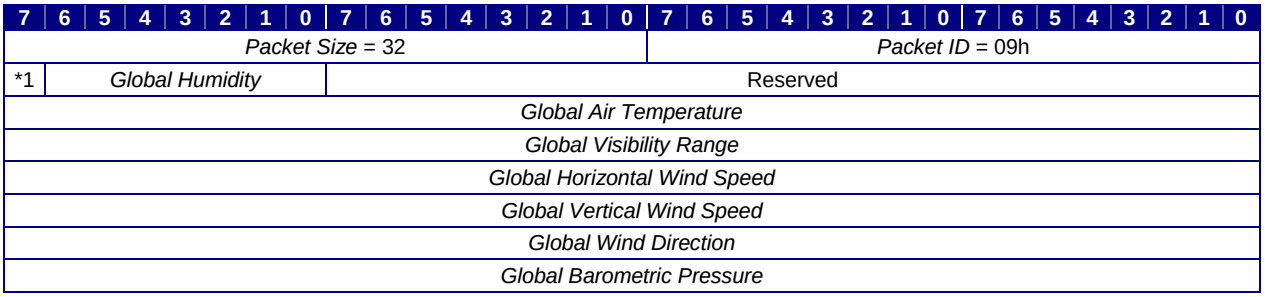
6.1.10 Atmosphere Control

The **Atmosphere Control** data packet allows the Host to control global atmospheric properties within the simulation.

Weather layers and weather entities shall always take precedence over the global atmospheric conditions. Once the atmospheric properties of a layer or entity are set, global atmospheric changes shall not affect the weather inside the layer or entity unless that layer or entity is disabled. The global atmospheric changes shall, however, affect the weather within a transition band or transition perimeter.

CIGI supports the parameters necessary to use Fast Atmospheric Signature Code (FASCODE) (H.J.P. Smith), MODerate resolution atmospheric TRANsmission (MODTRAN) (Spectral Sciences, Inc., 2012), SEDRIS (ISO/IEC, 2013), or other atmospheric models for determining radiance and transmittance within a heterogeneous atmosphere for sensor simulations. The *Atmospheric Model Enable* parameter determines whether an atmospheric model is used. The particular model is not specified and is determined by the IG.

The contents of the **Atmosphere Control** packet are as follows:



*1 *Atmospheric Model Enable*

Figure 53 – Atmosphere Control Packet Structure

Table 15 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 15 – Atmosphere Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 09h	This parameter identifies this data packet as the **Atmosphere Control** packet. The Host shall set this parameter to 09h.

Parameter	Description
Atmospheric Model Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	The value of this parameter specifies whether the IG shall use an atmospheric model to determine spectral radiances for sensor applications. If this parameter is set to Disable (0), source radiances shall be calculated. If this parameter is set to Enable (1), apparent radiances shall be calculated using the appropriate models.
Global Humidity Type: unsigned int8 Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall be applied as the global humidity of the environment.
Global Air Temperature Type: single float Units: degrees Celsius (°C) Default: IG-configurable	The value of this parameter shall be applied as the global air temperature of the environment.
Global Visibility Range Type: single float Units: meters Values: ≥ 0 Default: IG-configurable	The value of this parameter shall be applied as the global visibility range through the atmosphere.
Global Horizontal Wind Speed Type: single float Units: m/s Values: ≥ 0 Default: 0	The value of this parameter shall be applied as the global wind speed parallel to the ellipsoid-tangential reference plane.
Global Vertical Wind Speed Type: single float Units: m/s Default: 0	The value of this parameter specifies the global vertical wind speed. A positive value shall produce an updraft, while a negative value shall produce a downdraft.

Parameter	Description
Global Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	The value of this parameter specifies the global wind direction. This shall be the direction from which the wind is blowing.
Global Barometric Pressure Type: single float Units: millibars (mb) or hectopascals (hPa) Values: ≥ 0 Default: IG-configurable	The value of this parameter shall be applied as the global atmospheric pressure. The units are interchangeable.

6.1.11 Environmental Region Control

The **Environmental Region Control** packet is used to define an area over which the atmospheric conditions and maritime and terrestrial surface conditions may be specified. The shape of the region is a rounded rectangle, as shown below:

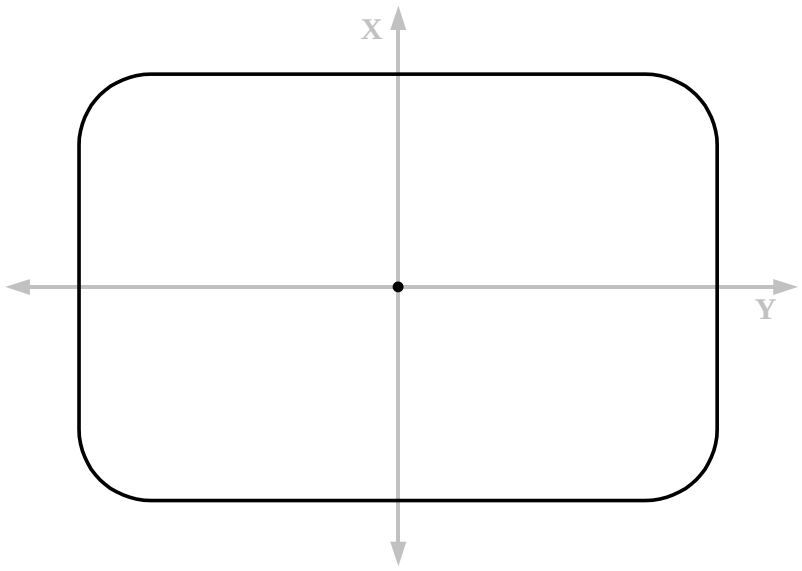


Figure 54 – Example of a Rounded Rectangle on NED Cartesian XY Plane

Up to 256 weather layers may be defined within a region. Weather layers shall be created and manipulated with the **Weather Control** packet (Section 6.1.12). One set of maritime and/or terrestrial surface condition parameters may be defined per region.

The Host is responsible for updating the position and shape of each region. The IG shall not automatically manipulate regions because of wind activity or any other internal or external forces.

The center of the region is defined by the *Latitude* and *Longitude* parameters. The origin of the region's local coordinate system is at this point. The *Size X* and *Size Y* parameters determine the length of the rounded rectangle along its **X** and **Y** axes (represented by **X´** and **Y´** in Figure 57), respectively.

The "roundness" of the corners is determined by the *Corner Radius* parameter. Setting this radius to zero (0) shall create a rectangle. Setting the value equal to one-half that of *Size X* and *Size Y* when both are equal shall create a circle. The corner radius shall be less than or equal to one half of the smaller of *Size X* or *Size Y*.

The *Rotation* parameter specifies an angle of rotation (clockwise) about the **Z** axis of the local NED coordinate system. Figure 55 shows a rounded rectangle on the NED reference plane rotated by a positive angle ψ.

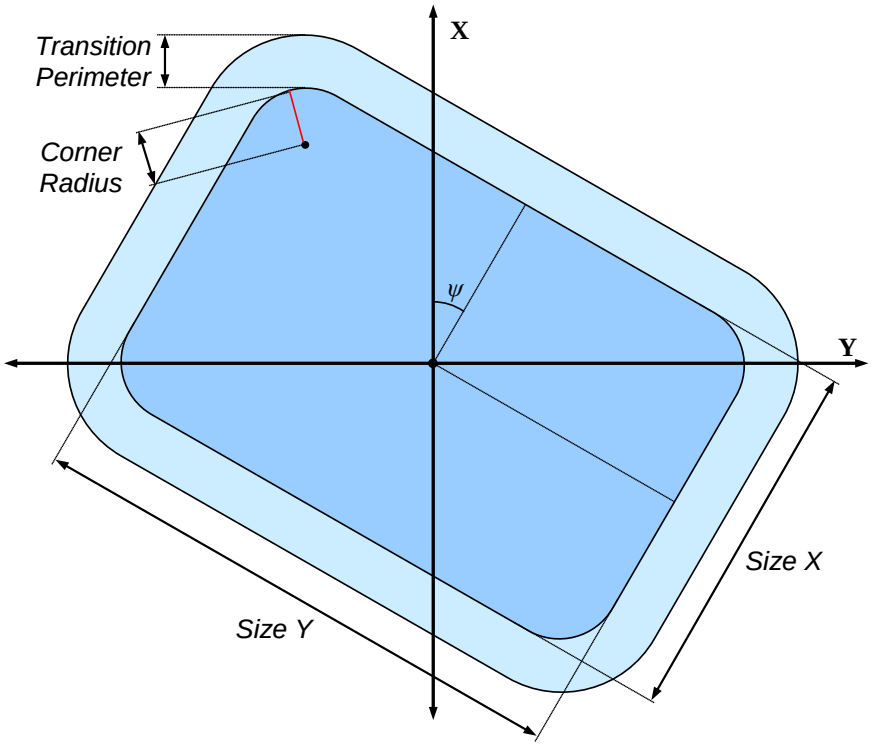


Figure 55 – Rotated Rounded rectangle with Transition Perimeter

The *Transition Perimeter* parameter specifies the width of a corridor around the outside of the rounded rectangle, as illustrated in Figure 55. Within this corridor, the weather conditions shall transition gradually from those inside the rounded rectangle to those immediately outside the corridor. This is analogous to the *Transition Band* parameter within the **Weather Control** packet (Section 6.1.12).

To determine the instantaneous value of a weather attribute at a point within the transition perimeter, the IG shall interpolate between the value inside the rounded rectangle and immediately outside the perimeter. For example, assume the temperature within the region, T_{region}, is defined to be 20°C and the global air temperature, T_{global}, is 40°C. The region has a transition perimeter width, p, of 1000m. The IG should perform interpolation to determine the temperature $T_{x,y}$ at some point (x, y) within the transition perimeter per Figure 56.

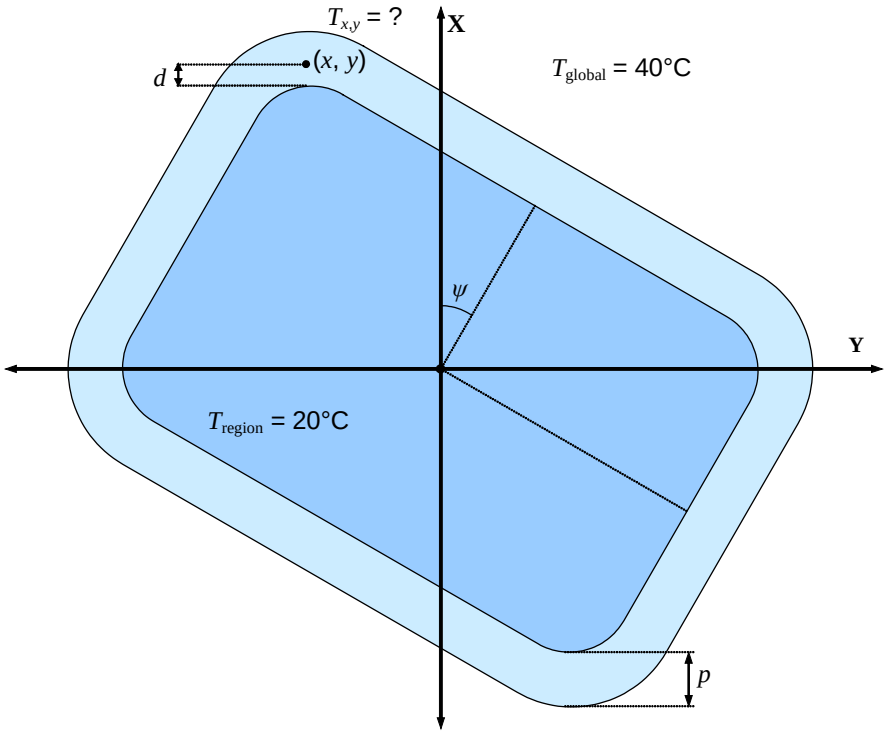


Figure 56 – Interpolation of Temperature within Transition Perimeter

In Figure 56, point (x, y) is some distance d from the edge of the rounded rectangle. This distance is measured along a line normal to the rounded rectangle. To determine whether this line emanates from one of the flat sides or one of the round corners, a coordinate transformation can be applied to (x, y) to determine the coordinates (x', y') of the point with respect to a coordinate system whose axes are parallel to the sides of the rectangle (see Figure 57). The values of x' and y' can be calculated from the following equations:

$$x' = y\sin\psi + x\cos\psi \quad (1)$$

$$y' = y\cos\psi - x\sin\psi \quad (2)$$

Figure 57 shows the rounded rectangle relative to the new coordinate system:

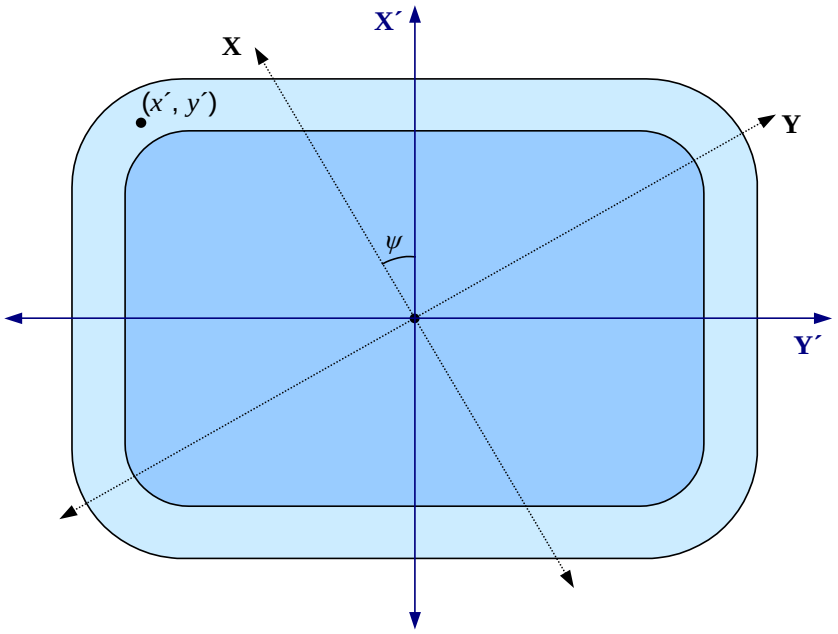


Figure 57 – Rounded Rectangle after Coordinate System Transformation

Determining whether the point is at a corner or along a straight side is now trivial.

Because point ($x´$, $y´$) in this example is at one of the corners, the distance d is measured along a line emanating from the corner's focal point (x_f, y_f). A circle with radius r centered at this focal point can be drawn through point ($x´$, $y´$) as shown in Figure 58:

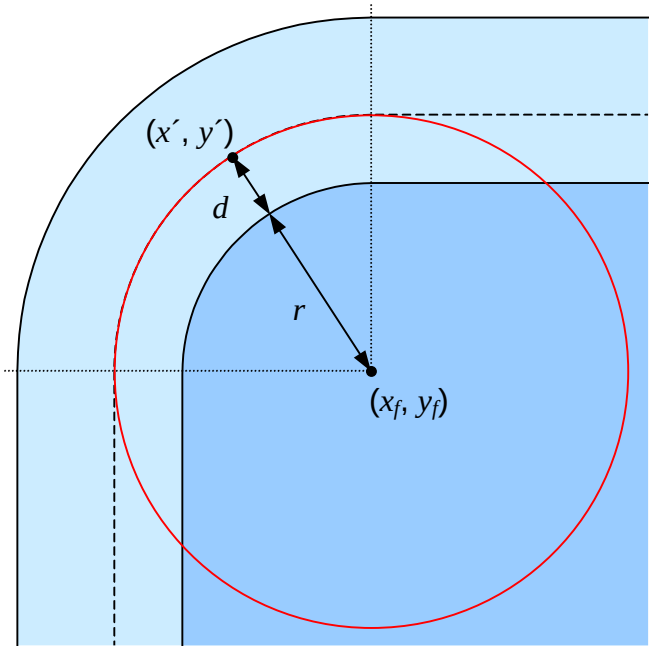


Figure 58 – Circle Drawn though Point (*x´*, *y´*)

Equations (3) and (4) give the coordinates of the focal point in the second quadrant:

$$x_f = r - \frac{(Size\ X)}{2} \tag{3}$$

$$y_f = \frac{(Size\ Y)}{2} - r \tag{4}$$

The value of d can be found from the equation for the circle:

$$d = \sqrt{(x' - x_f)^2 + (y' - y_f)^2} - r \tag{5}$$

The value of each attribute at point (x, y) can now be linearly interpolated. Continuing the example, the temperature at point (x, y) would be given by the following equation:

$$T_{x,y} = \frac{d\,(T_{\text{global}} - T_{\text{region}})}{p} + T_{\text{region}} \tag{6}$$

Should (x, y) be found to be 400 meters from the edge of the region, for instance, then the air temperature at that point would be calculated as follows:

$$T_{x,y} = \frac{400\text{m} \cdot (40°\text{C} - 20°\text{C})}{1000\text{m}} + 20°\text{C} = 28°\text{C} \tag{7}$$

Note that once d is found, the IG may use other functions for interpolating the values of weather attributes across a transition perimeter. The linear interpolation function shown by Equations (6) and (7) in the preceding example is used merely for illustration purposes.

Weather layers in one region may overlap layers in other regions. Similarly, any global weather layers shall overlap layers in other regions. Figure 59 shows two overlapping environmental regions, each containing a cloud layer. The cross-hatched areas indicate the region of overlap.

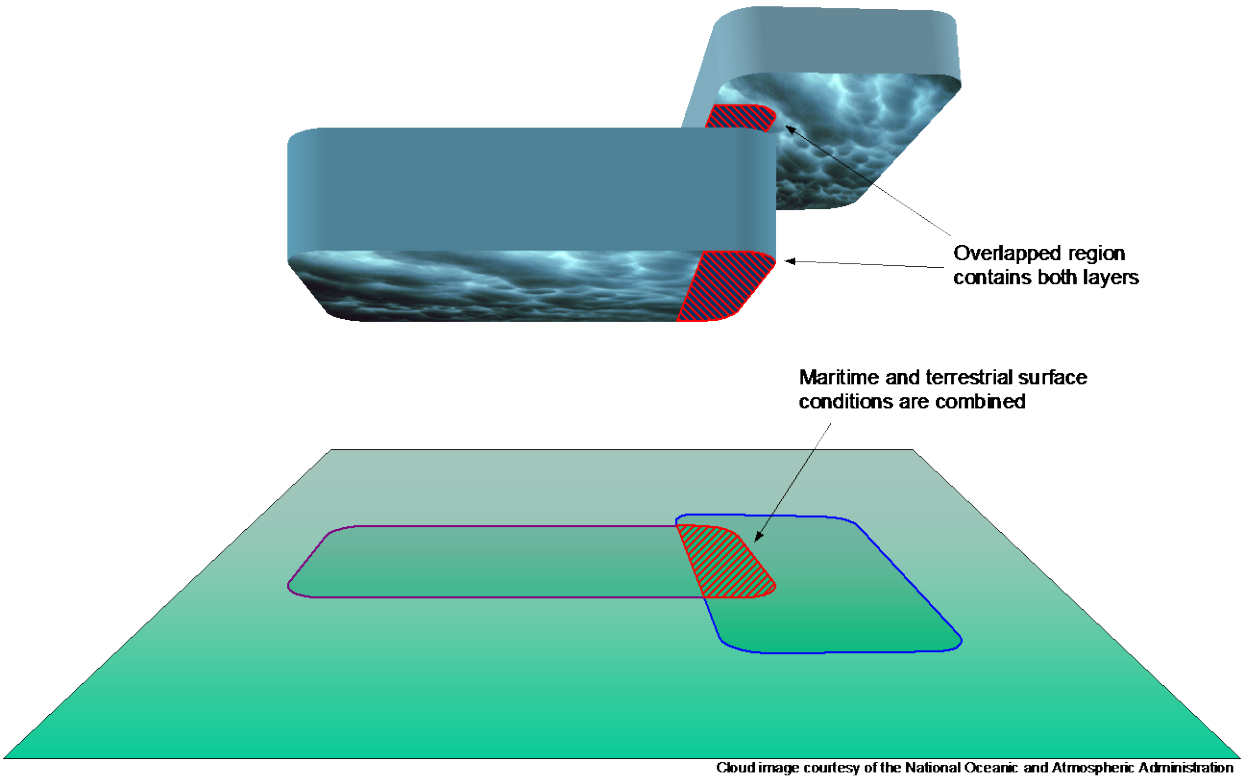


Figure 59 – Example of Overlapping Environmental Regions

If two overlapping regions contain layers that intersect, the values of each atmospheric property within the intersecting volumes may be combined. The *Merge Weather Properties* and *Merge Aerosol Concentrations* parameters shall determine whether the atmospheric properties and/or aerosol concentrations are combined for a given point within the volume or the last **Weather Control** packet describing the point is used. If these two parameters differ for two intersecting regions, priority shall be given to the region whose *Merge Weather Properties* or *Merge Aerosol Concentrations* parameter is set to Merge (1). Table 16 lists the recommended method of combining each property:

Table 16 – Recommended Methods of Combining Atmospheric Properties

Atmospheric Property	Recommended Function
Aerosol Concentration	weighted average (if same *Layer ID*) or blend (if different *Layer IDs*)
Air Temperature	weighted average
Barometric Pressure	maximum
Humidity	maximum
Visibility Range	minimum
Wind Velocity	sum of velocity vectors

Note that Table 16 lists two different methods for combining the *Aerosol Concentration* attribute of intersecting volumes. If the *Layer ID* of each layer is the same, the resulting aerosol concentration shall

be the average of the concentrations (taking into account interpolation through transition bands). This allows for complex regional shapes formed by combining two or more regions. If the *Layer ID* values assigned to the intersecting weather layers are different, then the aerosols are assumed to be different and shall each be present in their specified concentrations. Any visual or spectral effects caused by the aerosols shall be calculated independently for each aerosol.

Maritime and terrestrial surface conditions shall be combined within areas where regions overlap as defined in the **Maritime Surface Conditions Control** packet (Section 6.1.13) and **Terrestrial Surface Conditions Control** packet (6.1.15).

Multiple environmental regions may be arranged to form a weather grid. Figure 60 illustrates a portion of a grid created by adjacent rectangles, each with a narrow transition perimeter and a corner radius of zero. The color of each cell indicates the severity of some atmospheric phenomenon within, such as visibility range or wind speed. As an entity moves from one cell to another, it passes through two or more transition perimeters. The effect is a continuous intensity gradient rather than an abrupt change at cell boundaries.

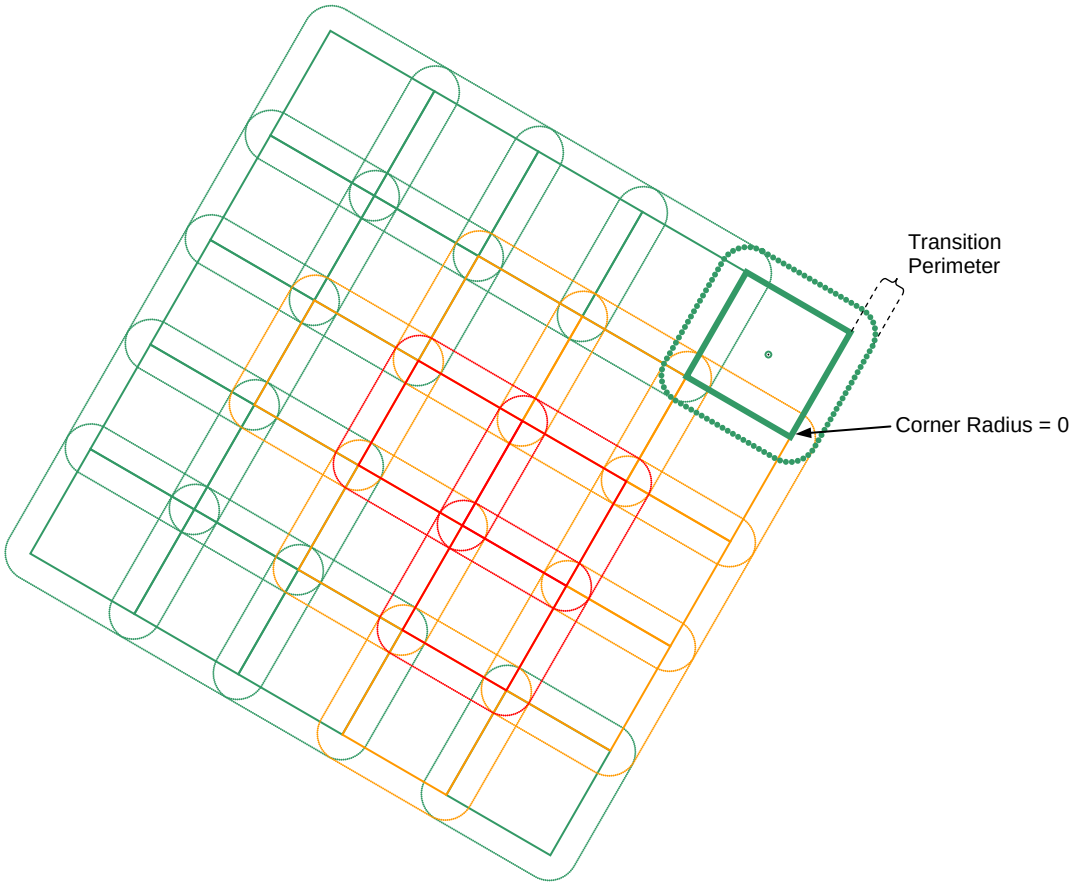


Figure 60 – Example of Gridded Weather System

Circular environmental regions may be used to approximate non-rectangular weather cells. These regions would have lengths and heights of zero and corner radii equal to the radius of each circle. A transition perimeter would ensure that an entity would experience a continuous, gradual transition at cell boundaries rather than a sudden change. Figure 61 illustrates a group of environmental regions approximating a system of hexagonal weather cells.

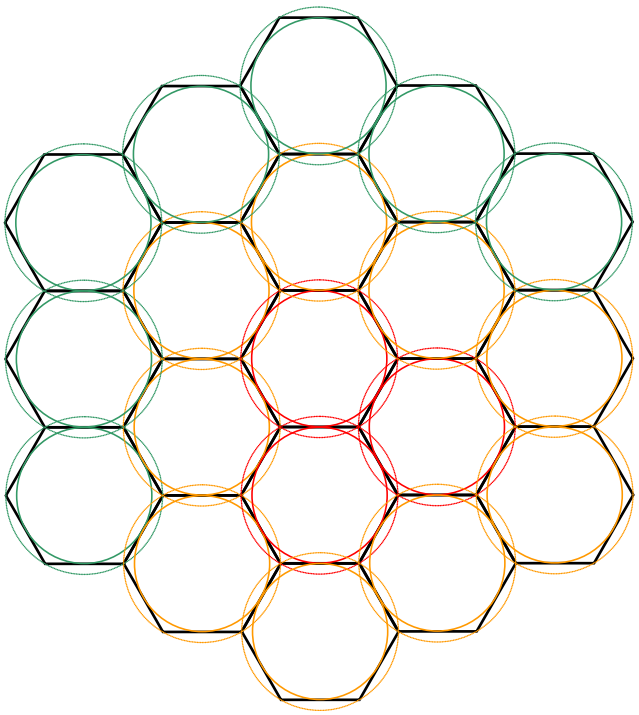


Figure 61 – Example of Approximation of Hexagonal Weather Cells

The contents of the **Environmental Region Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 48																*Packet ID* = 0Ah															
*6		*5		*4		*3		*2		*1		Reserved								*Region ID*											
Latitude																															
Longitude																															
Size X																															
Size Y																															
Corner Radius																															
Rotation																															
Transition Perimeter																															
Reserved																															

*1 *Region State*
*2 *Merge Weather Properties*
*3 *Merge Aerosol Concentrations*
*4 *Merge Maritime Surface Conditions*
*5 *Merge Terrestrial Surface Conditions*
*6 Reserved

Figure 62 – Environmental Region Control Packet Structure

Table 17 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 17 – Environmental Region Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 48.
Packet ID Type: unsigned int16 Units: N/A Value: 0Ah	This parameter identifies this data packet as the **Environmental Region Control** packet. The Host shall set this parameter to 0Ah.
Region State Type: unsigned 2-bit field Units: N/A Values: 0 Inactive 1 Active 2 Destroyed	The value of this parameter specifies whether the region may be inactive, active or destroyed. This parameter may be set to one of the following values: **Inactive** – Any weather layers and surface conditions defined within the region shall be disabled regardless of their individual enable states. **Active** – Any weather layers and surface conditions defined within the region shall be enabled according to their individual enable states. **Destroyed** – The environmental region shall be permanently deleted, as are all weather layers and surface conditions assigned to the region.

Parameter	Description
Merge Weather Properties Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	The value of this parameter specifies whether atmospheric conditions within this region shall be merged with those of other regions within areas of overlap. If this parameter is set to Use Last (0), the last **Weather Control** packet (Section 6.1.12) describing a layer containing a given point shall be used to determine the weather conditions at that point. If this parameter is set to Merge (1), the atmospheric properties of all weather layers containing a given point shall be combined (see Table 16). Note: Weather layers overlapping in altitude within the same region shall always be combined. Regional weather conditions always take priority over global weather conditions.
Merge Aerosol Concentrations Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	The value of this parameter specifies whether the concentrations of aerosols found within this region shall be merged with those of other regions within areas of overlap. If this parameter is set to Use Last (0), the last **Weather Control** packet (Section 6.1.12) describing a layer containing a given point shall be used to determine the concentration of the specified aerosol at that point. If this parameter is set to Merge (1), the aerosol concentrations within all weather layers containing a given point shall be combined (see Table 16). Note: Weather layers overlapping in altitude within the same region shall always be combined. Regional weather conditions always take priority over global weather conditions.

Parameter	Description
Merge Maritime Surface Conditions Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	The value of this parameter specifies whether the maritime surface conditions found within this region shall be merged with those of other regions within areas of overlap. If this parameter is set to Use Last (0), the last **Maritime Surface Conditions Control** packet (Section 6.1.13) describing a region containing a given point shall be used to determine the surface conditions at that point. If this parameter is set to Merge (1), the surface conditions at any given point within the region shall be averaged with those of any other regions also containing that point. Note: Regional surface conditions always take priority over global surface conditions.
Merge Terrestrial Surface Conditions Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	The value of this parameter specifies whether the terrestrial surface conditions found within this region shall be merged with those of other regions within areas of overlap. If this parameter is set to Use Last (0), the last **Terrestrial Surface Conditions Control** packet (Section 6.1.15) describing a region containing a given point shall be used to determine the surface conditions at that point. If this parameter is set to Merge (1), the surface conditions at any given point within the region shall be averaged with those of any other regions also containing that point. Note: Regional surface conditions always take priority over global surface conditions.
Region ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the environmental region to which the data in this packet shall be applied.
Latitude Type: double float Units: degrees Values: -90.0 – 90.0 Default: IG-configurable Datum: Equator	The value of this parameter shall be applied as the geodetic latitude of the center of the rounded rectangle.

Parameter	Description
Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Default: IG-configurable Datum: Prime Meridian	The value of this parameter shall be applied as the geodetic longitude of the center of the rounded rectangle.
Size X Type: single float Units: meters Values: > 0 Default: IG-configurable Datum: Ellipsoid-tangential reference plane	The value of this parameter shall be applied as the length of the environmental region along its **X** axis at the geoid surface. This length shall not include the width of the transition perimeter.
Size Y Type: single float Units: meters Values: > 0 Default: IG-configurable Datum: Ellipsoid-tangential reference plane	The value of this parameter shall be applied as the length of the environmental region along its **Y** axis at the geoid surface. This length shall not include the width of the transition perimeter.
Corner Radius Type: single float Units: meters Values: 0 to lesser of (½ × *Size X*) or (½ × *Size Y*) Default: IG-configurable	The value of this parameter shall be applied as the radius of the corner of the rounded rectangle. The smaller the radius, the “tighter” the corner. A value of 0.0 shall produce a rectangle.
Rotation Type: single float Units: degrees Values: -180.0 – 180.0 Default: IG-configurable Datum: True North	The value of this parameter shall be applied as the yaw angle of the rounded rectangle.

Parameter	Description
Transition Perimeter Type: single float Units: meters Values: ≥ 0 Default: IG-configurable	The value of this parameter shall be applied as the width of the transition perimeter around the environmental region. This perimeter is a region through which the weather conditions are interpolated between those inside the environmental region and those immediately outside the perimeter.

6.1.12 Weather Control

The **Weather Control** packet is used to control weather layers and weather entities. Global weather layers should have no distinct horizontal boundaries. Atmospheric affects shall be observed anywhere within the vertical range of the layer. Regional weather layers occur only in areas defined by the **Environmental Region Control** packet (Section 6.1.11). Weather entities are entities that represent meteorological phenomena.

The *Layer ID* parameter specifies the global or regional weather layer whose attributes are being set. If the *Scope* parameter is set to Global (0), the weather layer shall exist everywhere over the database. If this parameter is set to Region (1), the weather layer shall be bound to the region specified by the *Region ID* parameter. Up to 256 weather layers may be defined globally, and up to 256 layers may be defined within each region. The *Layer ID* parameter shall be ignored for weather entities.

The *Cloud Type* parameter specifies the type of cloud found within a cloud layer or entity. Each value shall correspond to a specific cloud texture or model. Values one through 10 are reserved for the most common general cloud types as listed in Table 18. The remaining values may be used for mammatus clouds, Kelvin-Helmholtz cloud effects, and other specific cloud phenomena.

The vertical range of a weather layer is specified by the *Base Elevation*, *Thickness*, and *Transition Band* parameters. *Base Elevation* specifies the distance from Mean Sea Level to the bottom of the layer. *Thickness* is the vertical height of the layer. *Transition Band* specifies the vertical height of both the region above and below the layer through which visibility gradually changes from that of the layer to that immediately outside the region. Figure 63 illustrates the use of these parameters:

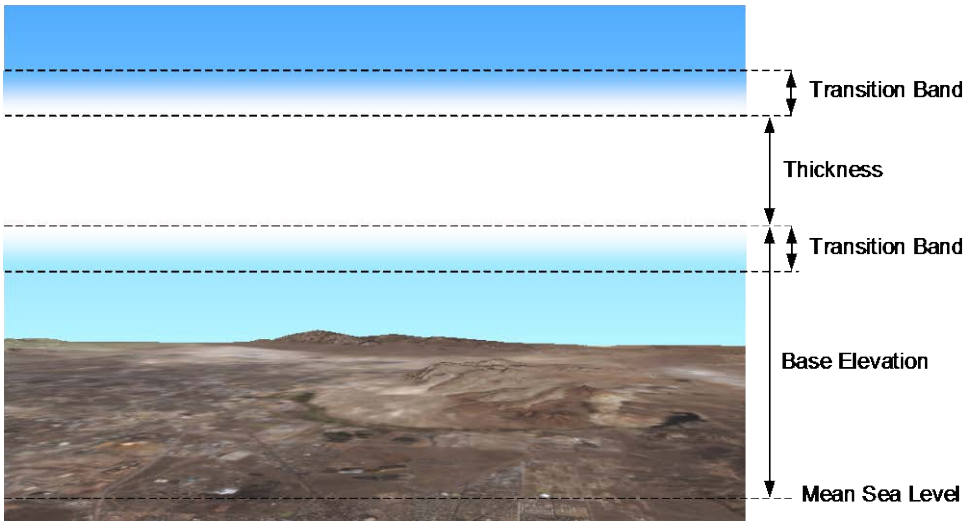


Figure 63 – Weather Layer Base Elevation and Thickness

For weather entities, the *Transition Band* parameter may be used to specify a non-planar transition region for partial penetration into a cloud model. The *Base Elevation* and *Thickness* parameters shall be ignored for weather entities.

The *Top* and *Bottom Scud Enable* parameters specify whether the layer produces scud effects within the top and bottom transition bands. The *Top* and *Bottom Scud Frequency* parameters define how often scud occurs.

The *Horizontal Wind Speed*, *Vertical Wind Speed*, and *Wind Direction* parameters define the wind velocity within the weather layer or entity. These may be used to specify surface winds or winds aloft, depending upon the base elevation and thickness of the layer or the altitude of the weather entity. The *Random Winds Enable* parameter shall cause the IG to create gusts of random duration and frequency.

A typical effect of weather layers is the suspension of liquid or solid particles in the air. The density of this particulate matter is specified by the *Aerosol Concentration* parameter. The most common aerosol is liquid water, but ice crystals, sand, and dust are also common. Weather layers may also be used to create smoke, combat haze, and other man-made airborne contaminants. Each layer may contain zero or one type of aerosol; multiple aerosols in a given space shall be implemented as separate weather layers.

Where weather layers overlap, atmospheric effects shall be combined as shown in Table 16 (page 112).

The contents of the **Weather Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 72</td><td colspan="16">Packet ID = 0Bh</td></tr>
<tr><td colspan="8">Layer ID</td><td colspan="8">Humidity</td><td>*5</td><td>*4</td><td>*3</td><td>*2</td><td>*1</td><td colspan="2">*9</td><td>*8</td><td colspan="2">*7</td><td colspan="2">*6</td></tr>
<tr><td colspan="16">Entity ID/Region ID</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="32">Air Temperature</td></tr>
<tr><td colspan="32">Visibility Range</td></tr>
<tr><td colspan="32">Bottom Scud Frequency</td></tr>
<tr><td colspan="32">Coverage</td></tr>
<tr><td colspan="32">Base Elevation</td></tr>
<tr><td colspan="32">Thickness</td></tr>
<tr><td colspan="32">Bottom Transition Band Thickness</td></tr>
<tr><td colspan="32">Horizontal Wind Speed</td></tr>
<tr><td colspan="32">Vertical Wind Speed</td></tr>
<tr><td colspan="32">Wind Direction</td></tr>
<tr><td colspan="32">Barometric Pressure</td></tr>
<tr><td colspan="32">Aerosol Concentration</td></tr>
<tr><td colspan="32">Top Scud Frequency</td></tr>
<tr><td colspan="32">Top Transition Band Thickness</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *Weather Enable*
*2 *Bottom Scud Enable*
*3 *Random Winds Enable*
*4 *Random Lightning Enable*
*5 *Cloud Type*
*6 *Scope*
*7 *Severity*
*8 *Top Scud Enable*
*9 Reserved

Figure 64 – Weather Control Packet Structure

Table 18 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 18 – Weather Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 72	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 72.
Packet ID Type: unsigned int16 Units: N/A Value: 0Bh	This parameter identifies this data packet as the **Weather Control** packet. The Host shall set this parameter to 0Bh.
Layer ID Type: unsigned int8 Units: N/A Values: 0 Ground Fog 1 Cloud Layer 1 2 Cloud Layer 2 3 Cloud Layer 3 4 Rain 5 Snow 6 Sleet 7 Hail 8 Sand 9 Dust 10 – 255 Defined by IG	The value of this parameter shall be applied as the weather layer to which the data in this packet are applied. This parameter also determines the type of aerosol contained within the layer. Values 0 through 9 are defined as standard weather layer types. Values beyond this range are defined in the IG configuration. Note: This parameter shall be ignored if *Scope* is set to Entity (2).
Humidity Type: unsigned int8 Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall be applied as the humidity within the weather layer/entity.
Weather Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether a weather layer/entity and its atmospheric effects are enabled.

Parameter	Description
Bottom Scud Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether the weather layer produces scud effects within its transition band located at the bottom of the weather layer.
Random Winds Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether a random frequency and duration shall be applied to the local wind effects.
Random Lightning Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether the weather layer or entity exhibits random lightning effects. The frequency and severity of the lightning shall vary according to the *Severity* parameter.
Cloud Type Type: unsigned 4-bit field Units: N/A Values: 0 None 1 Altocumulus 2 Altostratus 3 Cirrocumulus 4 Cirrostratus 5 Cirrus 6 Cumulonimbus 7 Cumulus 8 Nimbostratus 9 Stratocumulus 10 Stratus 11 – 15 Other Default: IG-configurable	The value of this parameter shall determine the type of clouds contained within the weather layer. If the value of *Layer ID* does not correspond to a cloud layer, *Cloud Type* shall be set to None (0).

Parameter	Description
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	The value of this parameter specifies whether the weather is global, regional, or assigned to an entity. If this value is set to Global (0), the layer shall be applied to the entire environment. If this value is set to Regional (1), the layer shall be confined to the region specified by *Region ID*. If this value is set to Entity (2), the weather attributes shall be applied to the volume within the moving model specified by *Entity ID*. Note: Regional and entity layer conditions override global layer conditions.
Severity Type: unsigned 3-bit field Units: N/A Values: 0 – 5 (Least to most severe) Default: IG-configurable	The value of this parameter shall determine the severity of the weather layer/entity.
Top Scud Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether the weather layer produces scud effects within its transition band located at the top of the weather layer.
Entity ID (Weather Entities) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Entity (2) the value of this parameter specifies the entity to which the packet data shall be applied.
Region ID (Regional Layers) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Regional (1) The value of this parameter specifies the region to which the packet data shall be applied. Note: *Entity ID/Region ID* shall be ignored if *Scope* is set to Global (0).

Parameter	Description
Air Temperature Type: single float Units: degrees Celsius (°C) Default: IG-configurable	The value of this parameter shall be applied as the temperature within the weather layer/entity.
Visibility Range Type: single float Units: meters Values: See description at right Default: IG-configurable	The value of this parameter shall be applied as the visibility range through the weather layer/entity. This might correspond to Runway Visibility Range through ground fog, for example. The range specified by this parameter shall take precedence over that specified by the *Global Visibility Range* parameter of the **Atmosphere Control** packet (see Section 6.1.10).
Bottom Scud Frequency Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall determine the frequency of scud within the transition band below a cloud or fog layer. A value of 0% shall produce no scud effect; 100% shall produce a solid effect.
Coverage Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall determine the amount of area coverage for the weather layer.
Base Elevation Type: single float Units: meters Default: IG-configurable Datum: Mean Sea Level	The value of this parameter shall be applied as the altitude of the base (bottom) of the weather layer. This parameter shall be ignored if *Scope* is set to Entity (2).

Parameter	Description
Thickness Type: single float Units: meters Default: IG-configurable Datum: Base (specified by *Base Elevation*)	The value of this parameter shall be applied as the vertical thickness of the weather layer. The altitude of the top of the layer is equal to this value plus that specified by *Base Elevation*. This parameter shall be ignored if *Scope* is set to Entity (2).
Bottom Transition Band Thickness Type: single float Units: meters Default: IG-configurable	The value of this parameter shall be applied as the height of a vertical transition band below the weather layer. This band produces a visibility gradient from the layer's visibility to that immediately outside the transition band. This parameter shall be ignored if *Scope* is set to Entity (2).
Horizontal Wind Speed Type: single float Units: m/s Values: ≥ 0 Default: 0	The value of this parameter shall be applied as the local wind speed parallel to the ellipsoid-tangential reference plane.
Vertical Wind Speed Type: single float Units: m/s Default: 0	The value of this parameter shall be applied as the local vertical wind speed. Note: A positive value shall produce an updraft, while a negative value shall produce a downdraft.
Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	The value of this parameter shall be applied as the local wind direction. Note: This shall be the direction from which the wind is blowing.

Parameter	Description
Barometric Pressure Type: single float Units: millibars (mb) or hectopascals (hPa) Values: ≥ 0 Default: IG-configurable	The value of this parameter shall be applied as the atmospheric pressure within the weather layer or entity. The units are interchangeable.
Aerosol Concentration Type: single float Units: g/m^3 Values: ≥ 0 Default: IG-configurable	The value of this parameter shall be applied as the concentration of water, smoke, dust, or other particles suspended in the air. This parameter is provided primarily for sensor applications; any visual effect is secondary and is IG- and layer-dependent. Note: The type of aerosol depends upon the layer ID of a weather layer, or the entity type of a weather phenomenon entity.
Top Scud Frequency Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall determine the frequency of scud within the transition band above a cloud or fog layer. A value of 0% shall produce no scud effect; 100% shall produce a solid effect.
Top Transition Band Thickness Type: single float Units: meters Default: IG-configurable	The value of this parameter shall be applied as the height of a vertical transition band above the weather layer. This band produces a visibility gradient from the layer's visibility to that immediately outside the transition band. This parameter shall be ignored if *Scope* is set to Entity (2).

6.1.13 Maritime Surface Conditions Control

The **Maritime Surface Conditions Control** packet is used to specify the surface behavior for seas and other bodies of water. This packet is used in conjunction with the **Weather Control** and **Wave Control** packets to define sea states.

Regional maritime surface conditions shall always take precedence over the global surface conditions. Once the surface conditions of a region are set, global changes shall not affect the surface conditions within that region unless it is disabled. Global changes shall, however, contribute to the conditions within a region's transition perimeter.

If two or more regions overlap, the value of each surface condition attribute defining the sea state within the area of overlap shall be the average of the values determined by overlapping the regions.

To determine the maritime surface conditions within areas of overlap or through a transition perimeter, the Host may request the conditions at a specific latitude and longitude by issuing an **Environmental Conditions Request** packet (Section 6.1.28). The Host may request the instantaneous height of the water surface at a specific latitude and longitude by sending a **HAT/HOT Request** packet (Section 6.1.24).

The contents of the **Maritime Surface Conditions Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 24</td><td colspan="16">Packet ID = 0Ch</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="2">*3</td><td>*2</td><td>*1</td><td colspan="4">Reserved</td><td colspan="16">Entity ID/Region ID</td></tr>
<tr><td colspan="32">Sea Surface Height</td></tr>
<tr><td colspan="32">Surface Water Temperature</td></tr>
<tr><td colspan="32">Surface Clarity</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *Surface Conditions Enable*
*2 *Whitecap Enable*
*3 *Scope*

Figure 65 – Maritime Surface Conditions Control Packet Structure

Table 19 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 19 – Maritime Surface Conditions Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 0Ch	This parameter identifies this data packet as the **Maritime Surface Conditions Control** packet. The Host shall set this parameter to 0Ch.
Surface Conditions Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter determines the state of the specified surface conditions. If this parameter is set to Disable (0), the surface conditions within the region or entity shall be the same as the global maritime surface conditions. If the parameter is set to Enable (1), the surface conditions shall be defined by this packet. This parameter shall be ignored if *Scope* is set to Global (0).
Whitecap Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter shall determine whether whitecaps are enabled.
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	The value of this parameter specifies whether this packet is applied globally, applied to a region, or assigned to an entity. If this value is set to Global (0), the surface condition properties shall be applied to all maritime surfaces present in the database. If this value is set to Regional (1), the surface condition properties shall be applied only within the region specified by *Region ID*. If this value is set to Entity (2), the properties shall be applied to the area defined by the moving model specified by *Entity ID*. Note: Regional and entity surface conditions override global surface conditions.

Parameter	Description
Entity ID (Entity-based Surface Conditions) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Entity (2) the value of this parameter specifies the entity to which the packet data shall be applied.
Region ID (Regional Surface Conditions) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Regional (1) The value of this parameter specifies the region to which the packet data shall be applied. Note: *Entity ID/Region ID* shall be ignored if *Scope* is set to Global (0).
Sea Surface Height Type: single float Units: meters Datum: Mean Sea Level Default: IG-configurable	The value of this parameter shall be applied as the height of the water above MSL at equilibrium. This parameter may also be used to specify the tide level within the surf zone.
Surface Water Temperature Type: single float Units: degrees Celsius (°C) Default: IG-configurable	The value of this parameter shall be applied as the water temperature at the surface.
Surface Clarity Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter specifies the clarity of the water at its surface. This is used to control the visual effect of the water's turbidity and sediment type. A value of 100% shall indicate pristine water. A value of 0% shall indicate extremely turbid water.

6.1.14 Wave Control

The **Wave Control** packet is used to specify the behavior of waves propagating across the surface of a body of water. Examples include simulated swells and wind chop.

The basic waveform is defined by a wave height, wavelength, period, and direction of propagation. Wave height refers to the vertical distance between the wave's crest and trough. The wavelength is the distance from one crest to the next or from one trough to the next. These wave properties are illustrated below:

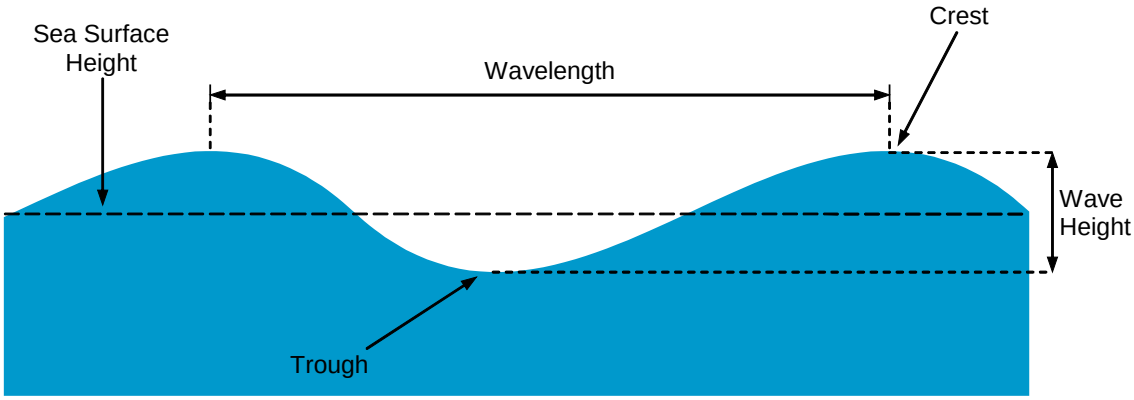


Figure 66 – Basic Wave Properties

The *Phase Offset* parameter specifies a phase angle to be added to the IG's reference phase. This is useful for modeling the interference patterns produced within a multiple-wave system.

The *Leading* parameter determines the cross-sectional shape of the wave. This value is the phase angle at which the crest of the wave occurs. For a sinusoidal wave, this angle is zero (0) degrees. As the value increases, the trough flattens and the crest moves toward the front of the wave as shown below:

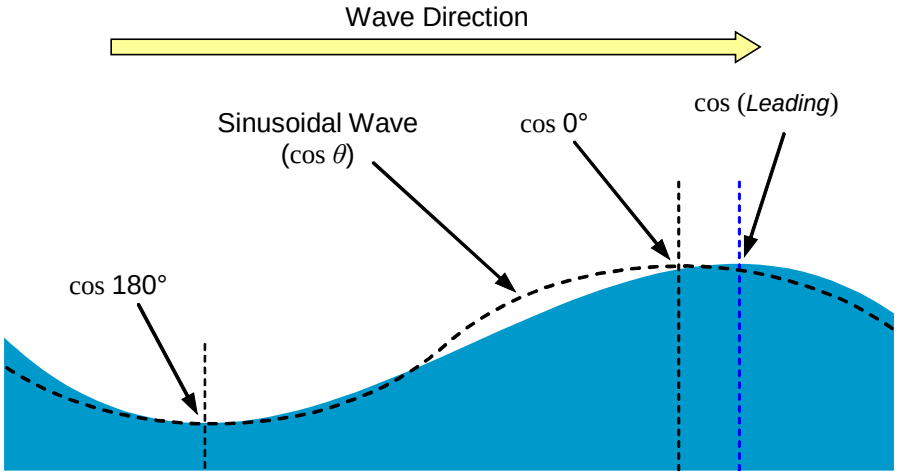


Figure 67 – Example of Wave Leading

Note that the trough of the wave remains at 180° regardless of the value of the *Leading* parameter.

The contents of the **Wave Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 0Dh															
Wave ID								Reserved				*3		*2		*1	*Entity ID/Region ID*															
Wave Height																															
Wavelength																															
Period																															
Direction																															
Phase Offset																															
Leading																															

*1 *Wave Enable*
*2 *Scope*
*3 *Breaker Type*

Figure 68 – Wave Control Packet Structure

Table 20 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 20 – Wave Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 0Dh	This parameter identifies this data packet as the **Wave Control** packet. The Host shall set this parameter to 0Dh.
Wave ID Type: unsigned int8 Units: N/A	The value of this parameter shall determine the wave to which the attributes in this packet are applied.
Wave Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter determines whether the wave is enabled or disabled. An enabled wave shall contribute to the shape of the water's surface. A disabled wave shall not contribute to the shape of the water's surface.

Parameter	Description
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	The value of this parameter specifies whether the wave is defined for global, regional, or entity-controlled maritime surface conditions. If this value is set to Global (0), the wave properties shall be applied to all water surfaces present in the database. If this value is set to Regional (1), the wave properties shall be applied only within the region specified by *Region ID*. If this value is set to Entity (2), the properties shall be applied to the area defined by the moving model specified by *Entity ID*. Note: Regional and entity wave conditions override global wave conditions.
Breaker Type Type: unsigned 2-bit field Units: N/A Values: 0 Plunging 1 Spilling 2 Surging Default: IG-configurable	The value of this parameter specifies the type of breaker within the surf zone. This may be one of the following values: **Plunging** – Plunging waves peak until the wave forms a vertical wall, at which point the crest moves faster than the base of the breaker. The wave shall then break violently into the wave trough. **Spilling** – Spilling breakers break gradually over a great distance. White water shall form over the crest, which spills down the face of the breaker. **Surging** – Surging breakers advance toward the beach as vertical walls of water. Unlike with plunging and spilling breakers, the crest shall not fall over the front of the wave.
Entity ID (Entity-based Surface Conditions) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Entity (2) the value of this parameter shall determine the entity to which the packet data shall be applied.
Region ID (Regional Surface Conditions) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Regional (1) The value of this parameter shall determine the region to which the packet data shall be applied. Note: *Entity ID/Region ID* shall be ignored if *Scope* is set to Global (0).

Parameter	Description
Wave Height Type: single float Units: meters Values: ≥ 0 Datum: *Sea Surface Height*	The value of this parameter shall be applied as the average vertical distance from trough to crest produced by the wave. (see Figure 66).
Wavelength Type: single float Units: meters Values: > 0	The value of this parameter shall be applied as the distance from a particular phase on a wave to the same phase on an adjacent wave. (see Figure 66).
Period Type: single float Units: seconds Values: > 0	The value of this parameter shall be applied as the time required for one complete oscillation of the wave.
Direction Type: single float Units: degrees Values: 0 – 360 Datum: True North	The value of this parameter shall be applied as the direction in which the wave propagates.
Phase Offset Type: single float Units: degrees Values: -360.0 – 360.0	The value of this parameter shall be applied as a phase offset for the wave.
Leading Type: single float Units: degrees Values: -180.0 – 180.0	This parameter specifies the phase angle at which the crest occurs (see Figure 67).

6.1.15 Terrestrial Surface Conditions Control

The **Terrestrial Surface Conditions Control** packet is used to specify the conditions of the terrain surface. These typically describe driving conditions, runway contaminants, or conditions that would otherwise impede or add risk to the movement of vehicles on the ground.

The possible surface conditions are IG-dependent. Examples might range from weather-related conditions such as dry, wet, icy, or slushy, to hazards such as sand, dirt, and gravel.

Regional terrestrial surface conditions shall take precedence over the global surface conditions. Once the surface conditions of a region are set, global changes shall not affect the surface condition within that region unless it is disabled. Global changes shall, however, change the conditions within a region's transition perimeter.

If two or more regions overlap, the value of each surface condition attribute within the area of overlap shall be the average of the values determined by the overlapping regions.

To determine the terrestrial surface conditions within areas of overlap or through a transition perimeter, the Host may request the conditions at a specific latitude and longitude by issuing an **Environmental Conditions Request** packet (Section 6.1.28).

The contents of the **Terrestrial Surface Conditions Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 16																*Packet ID* = 0Eh															
Entity ID/Region ID																*Severity*				*2		*1		*Coverage*							
Surface Condition ID																Reserved															
Reserved																															

*1 *Surface Condition Enable*
*2 *Scope*

Figure 69 – Terrestrial Surface Conditions Control Packet Structure

Table 21 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 21 – Terrestrial Surface Conditions Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 16.
Packet ID Type: unsigned int16 Units: N/A Value: 0Eh	This parameter identifies this data packet as the **Terrestrial Surface Conditions Control** packet. The Host shall set this parameter to 0Eh.

Parameter	Description
Entity ID (Environmental Entities) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Entity (2) the value of this parameter specifies the environmental entity to which the packet data shall be applied.
Region ID (Regional Conditions) Type: unsigned int16 Units: N/A	When the *Scope* field is set to Regional (1) The value of this parameter specifies the environmental region to which the packet data shall be applied. Note: *Entity ID/Region ID* shall be ignored if *Scope* is set to Global (0).
Surface Condition Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter specifies whether the surface condition attribute identified by the *Surface Condition ID* parameter shall be enabled.
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	The value of this parameter determines whether the specified surface conditions are applied globally, regionally, or to an environmental entity. If this value is set to Global (0), the conditions shall be applied to all appropriate surfaces present in the database. If this value is set to Regional (1), the conditions shall be confined to the region specified by *Region ID*. If this value is set to Entity (2), the conditions shall be applied to the model specified by *Entity ID*. Note: Regional and entity surface conditions override global surface conditions.
Severity Type: unsigned 5-bit field Units: N/A Values: 0 – 31 (least to most severe) Default: IG-configurable	The value of this parameter determines the degree of severity for the specified surface contaminant(s). A value of zero (0) shall indicate that any effects of the contaminant are negligible. A value of 31 shall indicate that the surface is not navigable.

Parameter	Description
Coverage Type: unsigned int8 Units: percent Values: 0 – 100 Default: IG-configurable	The value of this parameter shall determine the degree of coverage of the specified surface contaminant.
Surface Condition ID Type: unsigned int16 Units: N/A Values: 0 Dry (reset) > 0 Defined by IG Default: IG-configurable	The value of this parameter identifies a surface condition or contaminant. Multiple conditions may be specified by sending multiple **Terrestrial Surface Conditions Control** packets. When this parameter is set to Dry (0), all existing surface conditions shall be removed within the specified scope. When this parameter is set to other than Dry (0), all affected surfaces shall exhibit the corresponding IG defined surface conditions.

6.1.16 View Control

The **View Control** packet is used to attach a view or view group to an entity and to define the position and rotation of the view relative to the entity's reference point. Views may be positioned to correspond to the pilot eye, weapon/sensor viewpoints, and stealth view cameras.

Multiple views may be combined to form one or more view groups. This allows more than one view to be moved in unison with a single **View Control** packet. A view group is identified by the *Group ID* parameter. Operations performed upon a view group affect all views in that group. If *Group ID* is set to zero (0), the packet is applied to an individual view, identified by the *View ID* parameter.

The order of operation for views and view groups is the same as that for entities. A view shall first be translated along the entity's **X**, **Y**, and **Z** axes. After it is translated, the view is rotated about the eyepoint. The order of rotation shall first be about the **Z** axis (yaw), then the **Y** axis (pitch), and finally the **X** axis (roll). Figure 70 illustrates the degrees of freedom for positioning and rotating a view:

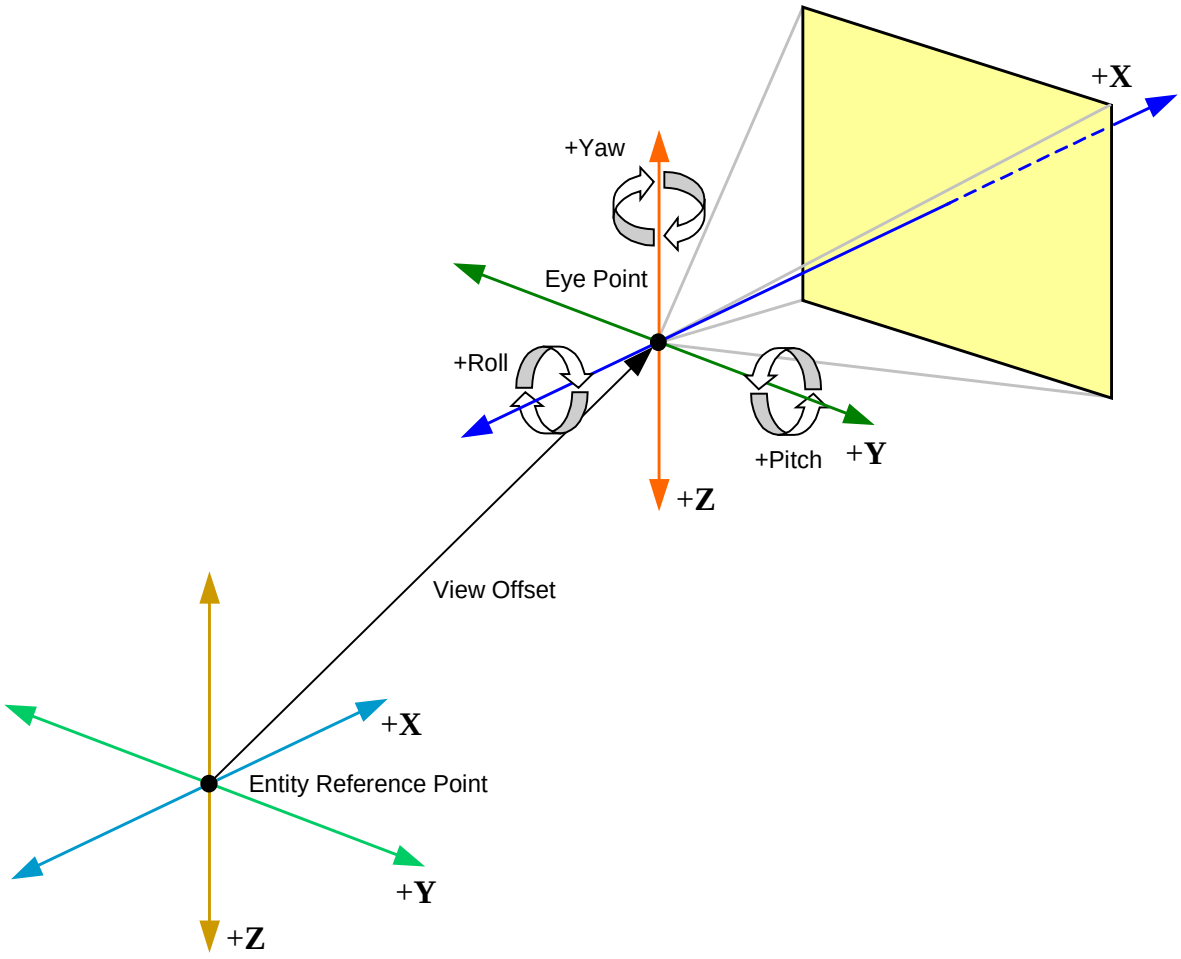


Figure 70 – View Point Position and Rotation

The contents of the **View Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 40																*Packet ID* = 0Fh															
Group ID								*7	*6	*5	*4	*3	*2	*1		*View ID*															
Entity ID																Reserved															
X Offset																															
Y Offset																															
Z Offset																															
Roll																															
Pitch																															
Yaw																															
Reserved																															

*1 *X Offset Enable*
*2 *Y Offset Enable*
*3 *Z Offset Enable*
*4 *Roll Enable*
*5 *Pitch Enable*
*6 *Yaw Enable*
*7 Reserved

Figure 71 – View Control Packet Structure

Table 22 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 22 – View Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 40.
Packet ID Type: unsigned int16 Units: N/A Value: 0Fh	This parameter identifies this data packet as the **View Control** packet. The Host shall set this parameter to 0Fh.

Parameter	Description
Group ID Type: unsigned int8 Units: N/A Values: 0 None 1 – 255 Specifies view group	The value of this parameter specifies the view group to which the contents of this packet are applied. If this value is zero (0), the packet shall be applied to the individual view specified by the *View ID* parameter. If this value is non-zero, the packet shall be applied to the specified view group and the *View ID* parameter shall be ignored.
X Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *X Offset* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *X Offset* parameter shall be applied. If this flag is set to Disable (0), the *X Offset* parameter shall be ignored.
Y Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Y Offset* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *Y Offset* parameter shall be applied. If this flag is set to Disable (0), the *Y Offset* parameter shall be ignored.
Z Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Z Offset* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *Z Offset* parameter shall be applied. If this flag is set to Disable (0), the *Z Offset* parameter shall be ignored.
Roll Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Roll* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *Roll* parameter shall be applied. If this flag is set to Disable (0), the *Roll* parameter shall be ignored.

Parameter	Description
Pitch Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Pitch* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *Pitch* parameter shall be applied. If this flag is set to Disable (0), the *Pitch* parameter shall be ignored.
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter determines whether the *Yaw* parameter shall be applied to the specified view or view group. If this flag is set to Enable (1), the *Yaw* parameter shall be applied. If this flag is set to Disable (0), the *Yaw* parameter shall be ignored.
View ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the view to which the contents of this packet shall be applied. This value shall be ignored if the *Group ID* parameter contains a non-zero value.
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the packet data shall be applied. Entity ID shall be ignored if the view is in a group.
X Offset Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the position of the view eyepoint along the **X** axis of the entity specified by the *Entity ID* parameter.
Y Offset Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the position of the view eyepoint along the **Y** axis of the entity specified by the *Entity ID* parameter.

Parameter	Description
Z Offset Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the position of the view eyepoint along the **Z** axis of the entity specified by the *Entity ID* parameter.
Roll Type: single float Units: degrees Values: -180 – 180 Default: IG-configurable Datum: View coordinate system	The value of this parameter shall be applied as the angle of rotation of the view or view group about its **X** axis after yaw and pitch have been applied.
Pitch Type: single float Units: degrees Values: -90 – 90 Default: IG-configurable Datum: View **XY** plane	The value of this parameter shall be applied as the angle of rotation of the view or view group about its **Y** axis after yaw has been applied.
Yaw Type: single float Units: degrees Values: 0 – 360 Default: IG-configurable Datum: View reference coordinate system	The value of this parameter shall be applied as the angle of rotation of the view or view group about its **Z** axis.

6.1.17 Sensor Control

The **Sensor Control** packet is used to control sensor modes and display behavior for sensor-based weapons systems and other sensor applications. It is typically used in conjunction the **View Control** packet (Section 6.1.16), which moves the sensor camera eyepoint. The **View Definition** and **Component Control** packets (Sections 6.1.21 and 6.1.4, respectively) may also be used to control various aspects of camera and sensor behavior.

A sensor is associated with a view through the *View ID* parameter. A sensor may be associated with more than one view to allow the sensor imagery to be displayed on multiple displays; however, this may evoke multiple **Sensor Response** or **Sensor Extended Response** packets from the IG.

The sensor shall be inactive until the Host turns the sensor on. The Host will send a **Sensor Control** packet with the *Sensor On/Off* parameter set to On (1). Because the sensor is not yet tracking a target, the *Track Mode* parameter of this packet should be set to Off (0). The Host might also send a **View Control** packet to make sure the initial sensor camera position is set. Additional **View Control** packets should be sent as the user slews the sensor view. This sequence of events is illustrated in Figure 72:

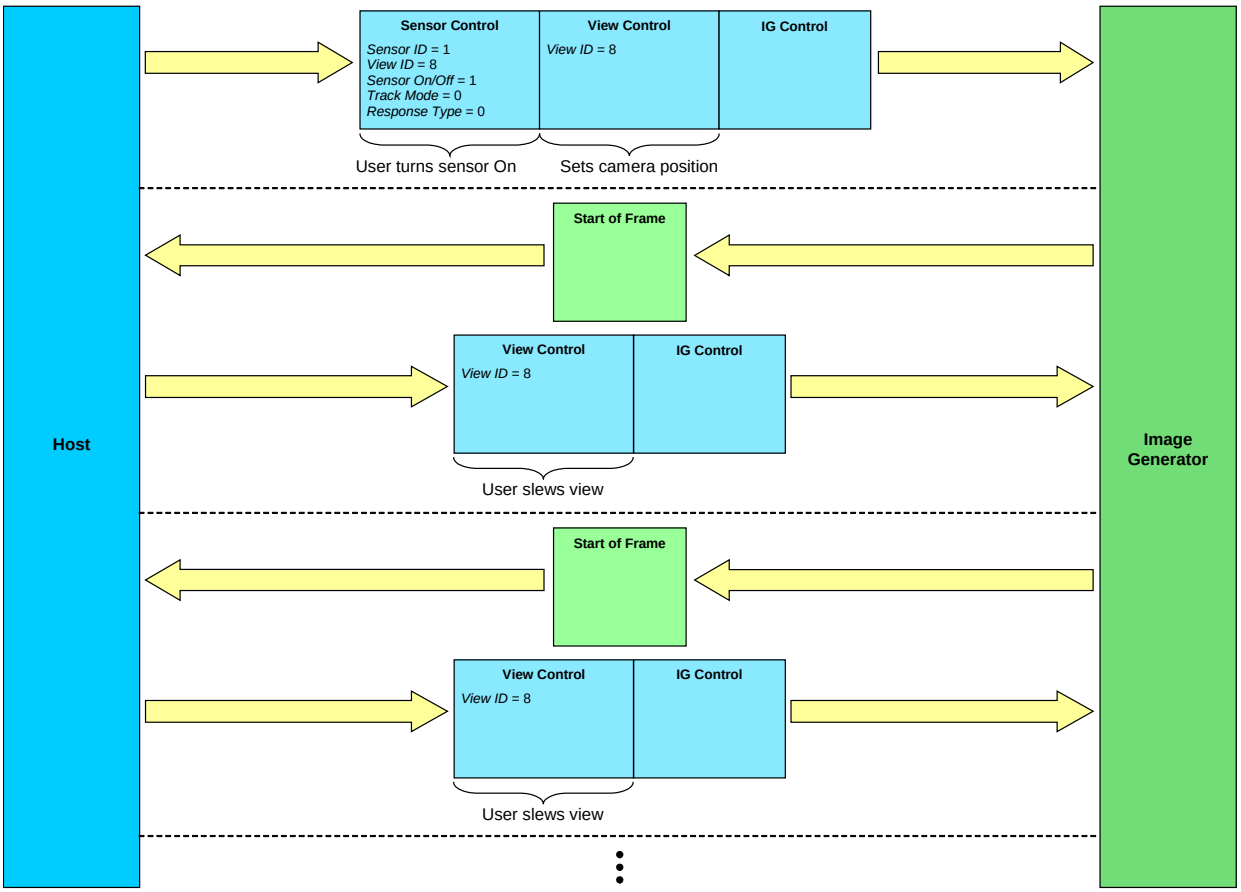


Figure 72 – Data Exchange for Sensor Control (1 of 3)

When the user attempts to lock onto a target, the Host shall send a **Sensor Control** packet, setting the *Track Mode* parameter to the appropriate value. Because the Host needs the position of the track point to determine which entity is the target, it sets the *Response Type* parameter to Gate and Target Position (1).

The IG should immediately begin sending response packets (in this case, **Sensor Extended Response** packets) that contain the gate symbol position and, if appropriate, the sensor target position. The IG response should be immediate to assure proper and uniform tracker slewing. A response packet should

be sent every frame with properly refreshed information until the IG is directed to do otherwise by the Host.

The *Sensor Status* parameter of the response packets indicates whether the sensor was able to establish a lock. If the sensor was unable to do so, the *Sensor Status* parameter shall be set to zero (0). The Host then should reset the *Track Mode* parameter to Off (0) before the user again tries to lock onto the target. On the other hand, if the lock was successful, then the *Sensor Status* parameter shall be set to one (1).

Figure 73 continues the example illustrated above. Here, the user attempts to acquire a sensor lock, prompting the Host to send a **Sensor Control** packet. The *Track Mode* parameter is set to Target (3) and the *Response Type* parameter to Gate and Target Position (1). The IG responds with **Sensor Extended Response** packets that indicate the lock was successful and provide gate and target positions.

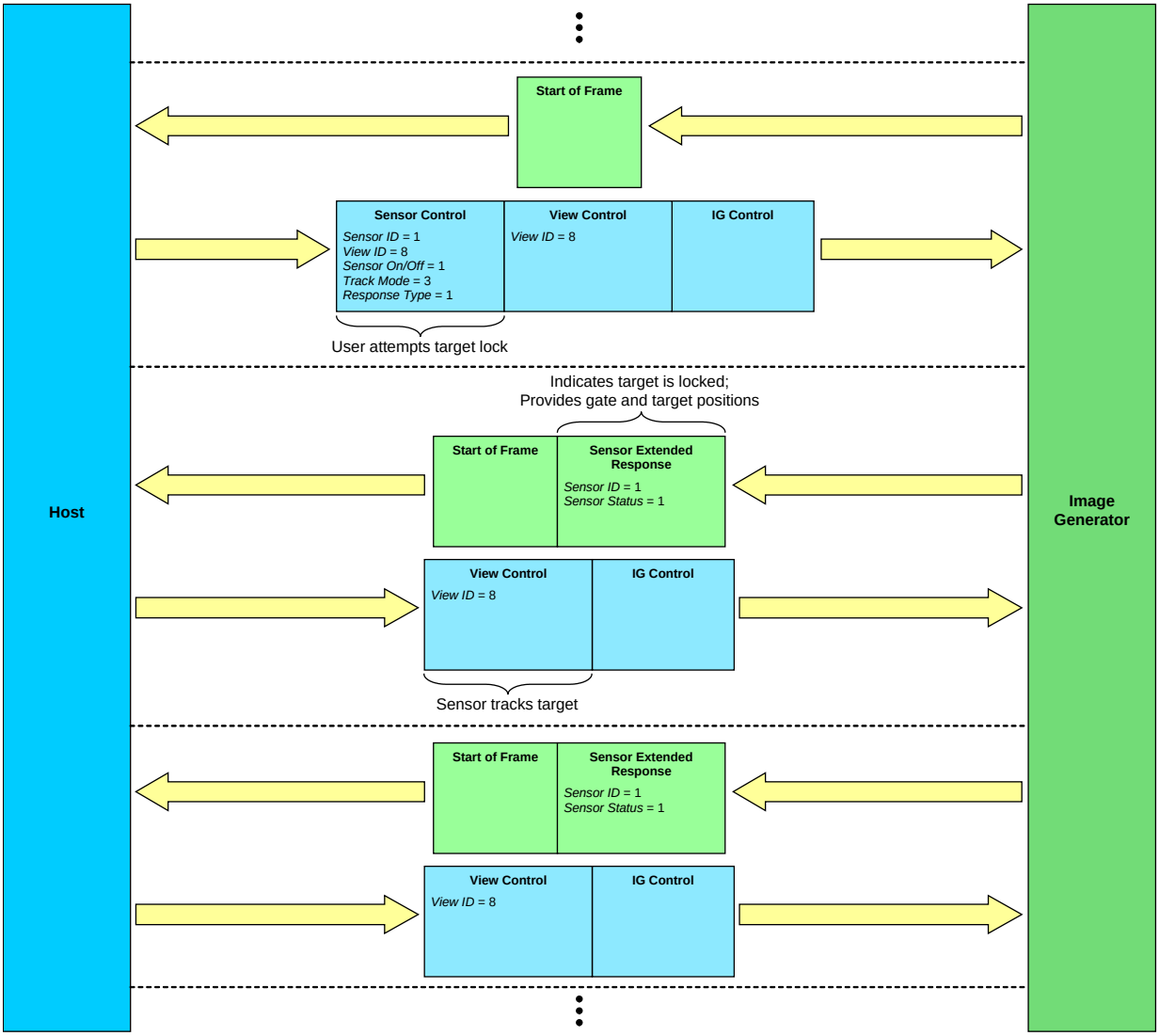


Figure 73 – Data Exchange for Sensor Control (2 of 3)

The *Entity ID* parameter of the **Sensor Extended Response** packet contains the ID of the target entity. If the IG is not able to determine the target, or if the sensor is tracking non-entity geometry, then the *Entity*

ID Valid parameter of the response packet shall be set to Invalid (0). The Host shall then use the target position returned by the IG to determine which entity or object is being tracked by the sensor.

Once the Host has determined the target bandwidth and processing overhead may be conserved, it should send a **Sensor Control** packet with its *Response Type* parameter set to Gate Position (1), thereby directing the IG to send **Sensor Response** packets instead of **Sensor Extended Response** packets. This exchange of data is illustrated in Figure 74:

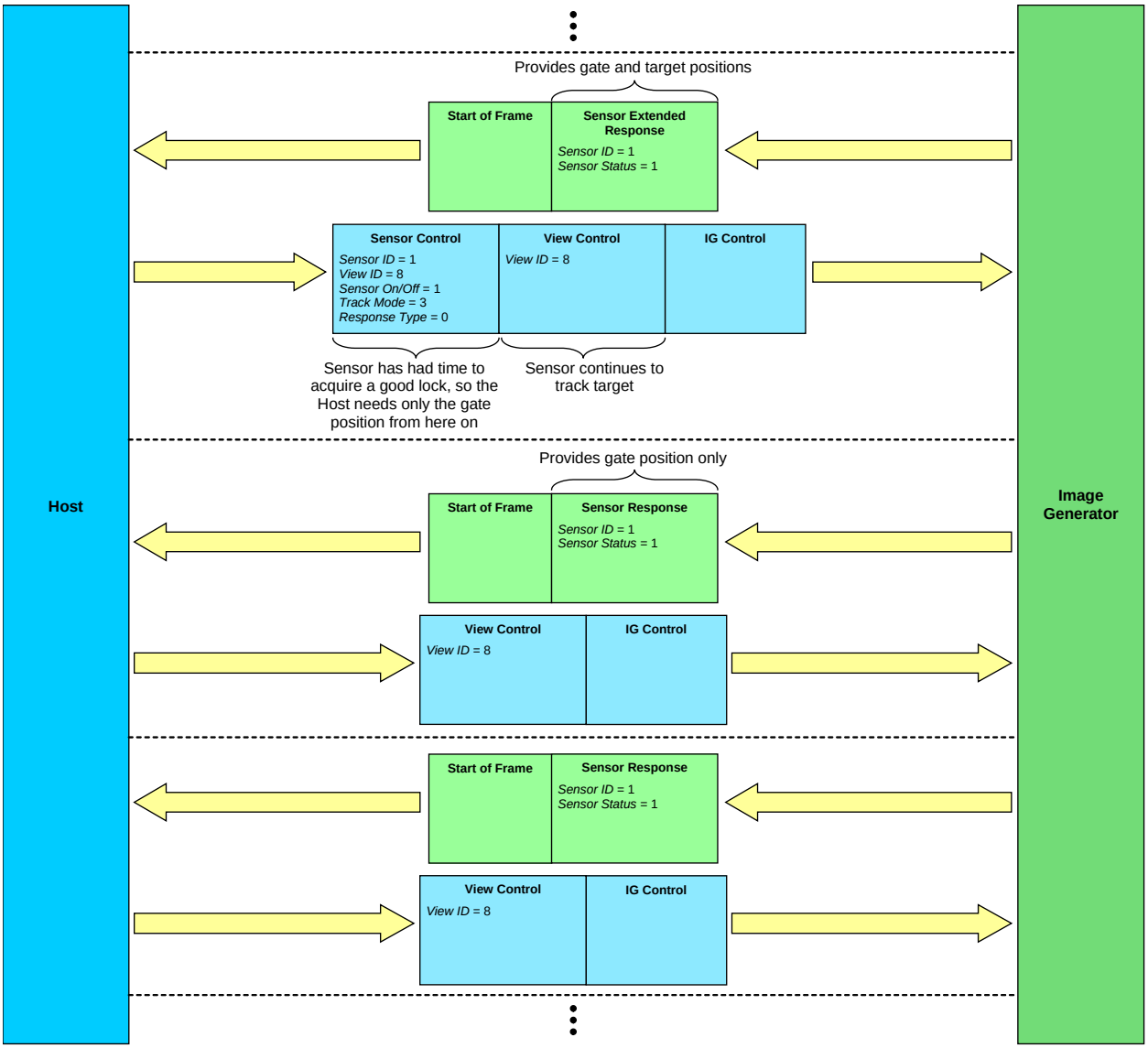


Figure 74 – Data Exchange for Sensor Control (3 of 3)

The contents of the **Sensor Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 010h															
Sensor ID								*7		*5		*4		*3		*2		*1	Reserved								*6	Reserved			
View ID																			Reserved												
Gain																															
Level																															
AC Coupling																															
Noise																															
Reserved																															

*1 *Sensor On/Off*
*2 *Polarity*
*3 *Line-by-Line Dropout Enable*
*4 *Automatic Gain*
*5 *Track White/Black*
*6 *Response Type*
*7 *Track Mode*

Figure 75 – Sensor Control Packet Structure

Table 23 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 23 – Sensor Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 010h	This parameter identifies this data packet as the **Sensor Control** packet. The Host shall set this parameter to 010h.
Sensor ID Type: unsigned int8 Units: N/A	The value of this parameter specifies the sensor to which the data in this packet shall be applied.

Parameter	Description
Track Mode Type: unsigned 3-bit field Units: N/A Values: 0 Off 1 Force Correlate 2 Scene 3 Target 4 Ship 5 – 7 Defined by IG Default: 0	The value of this parameter specifies which track mode the sensor may use: **Off** – When Track Mode is set to this value, no tracking shall occur. **Force Correlate** – When Track Mode is set to this value, the tracking behavior shall be similar to that found in a Maverick missile. Although the goal of the tracking is similar to Scene Mode the implementer should be aware that there are subtle differences between the two modes. However in general the sensor processes a portion of the view image, establishes an image pattern, and attempts to keep the seeker pointed at the center of that image pattern **Scene** – When Track Mode is set to this value, the tracking behavior shall be similar to that found in a FLIR (Forward Looking Infrared). Although the goal of the tracking is similar to Force Correlate Mode the implementer should be aware that there are subtle differences between the two modes. However in general the sensor processes a portion of the view image, establishes an image pattern, and attempts to keep the seeker pointed at the center of that image pattern. **Target** – When Track Mode is set to this value, contrast tracking shall be used to lock to a specific target area. **Ship** – When Track Mode is set to this value, contrast tracking shall be used, with adjustment of the tracking point so that the weapon strikes close to the water line.
Sensor On/Off Type: 1-bit field Units: N/A Values: 0 Off 1 On Default: 0	The value of this parameter shall determine whether the sensor is turned on or off.

Parameter	Description
Polarity Type: 1-bit field Units: N/A Values: 0 White hot 1 Black hot Default: 0	The value of this parameter shall determine whether the sensor shows white hot or black hot.
Line-by-Line Dropout Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	The value of this parameter shall determine whether line-by-line dropout is enabled. This effect is meant to simulate the horizontal stripes caused by a transient loss of video information.
Automatic Gain Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	The value of this parameter shall determine whether the sensor automatically adjusts the gain value to optimize the brightness and contrast of the sensor display.
Track White/Black Type: 1-bit field Units: N/A Values: 0 White 1 Black Default: 0	The value of this parameter shall determine whether the sensor tracks white or black. This, along with the *Polarity* parameter, controls whether the sensor tracks hot or cold spots.

Parameter	Description
Response Type Type: unsigned 1-bit field Units: N/A Values: 0 Normal (gate position) 1 Extended (gate and target position) Default: 0	The value of this parameter shall determine the type of response sent by the IG. If Response Type is set to Normal, the IG shall return a **Sensor Response** packet (Section 6.2.6). If Response Type is set to Extended, the IG shall return a **Sensor Extended Response** packet (Section 6.2.7). The IG shall return one of the two sensor response packets every frame as long as the following two criteria are met: 1. *Sensor On/Off* is set to On (1). 2. *Track Mode* is **not** set to Off (0).
View ID Type: unsigned int16 Units: N/A	The value of this parameter identifies the view to which the specified sensor shall be assigned. A sensor shall not be assigned to a view group.
Gain Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	The value of this parameter shall be applied as the contrast for the sensor display.
Level Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	The value of this parameter shall be applied as the brightness for the sensor display.
AC Coupling Type: single float Units: µs Values: ≥ 0.0 Default: 0.0	The value of this parameter shall be applied as the AC coupling decay constant for the sensor display.

Parameter	Description
Noise Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	The value of this parameter shall be applied as the amount of detector noise for the sensor.

6.1.18 Motion Tracker Control

The **Motion Tracker Control** packet is used to initialize and change properties of tracked input devices connected to the IG. These devices may include head trackers, eye trackers, wands, trackballs, etc. If more than one head tracker is used to control a view or view group, the order in which the transformations are applied shall be determined by the IG.

The Host may request the instantaneous position and orientation of a tracker device by sending a **Position Request** packet (Section 6.1.27) with its *Object Class* parameter set to Motion Tracker (4).

Note that if tracked input devices are connected to the Host, the Host should interpret the tracked input data and send the appropriate CIGI packets to achieve the desired effect on the IG. For example, the Host would interpret input from a connected head tracker and send **View Control** packets to the IG to move the eyepoint of the appropriate view or view group.

The contents of the **Motion Tracker Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 16																*Packet ID* = 011h															
Tracker ID								*8	*7	*6	*5	*4	*3	*2	*1	Reserved								*9	Reserved						
View/View Group ID																Reserved															
Reserved																															

*1 *Tracker Enable*
*2 *Boresight Enable*
*3 *X Enable*
*4 *Y Enable*
*5 *Z Enable*
*6 *Roll Enable*
*7 *Pitch Enable*
*8 *Yaw Enable*
*9 *View/View Group Select*

Figure 76 – Motion Tracker Control Packet Structure

Table 24 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 24 – Motion Tracker Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 16.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 011h	This parameter identifies this data packet as the **Motion Tracker Control** packet. The Host shall set this parameter to 011h.
Tracker ID Type: unsigned int8 Units: N/A	The value of this parameter shall determine the tracker whose state the data in this packet represents.
Tracker Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter specifies whether the tracking device shall be enabled.
Boresight Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	The value of this parameter shall set the boresight state of the external tracking device. This mode is used to reestablish the tracker's "center" position at the current position and orientation. Note: If boresighting is enabled, the Host shall send a **Motion Tracker Control** packet with *Boresight Enable* set to Disable (0) to return the tracker to normal operation. The IG shall continue to update the boresight position each frame until that occurs.
X Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the **X**-axis position of the motion tracker. If this flag is set to Enable (1), the motion tracker's **X**-axis positioning shall be applied. If this flag is set to Disable (0), the motion tracker's **X**-axis positioning shall be ignored.

Parameter	Description
Y Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the **Y**-axis position of the motion tracker. If this flag is set to Enable (1), the motion tracker's **Y**-axis positioning shall be applied. If this flag is set to Disable (0), the motion tracker's **Y**-axis positioning shall be ignored.
Z Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the **Z**-axis position of the motion tracker. If this flag is set to Enable (1), the motion tracker's **Z**-axis positioning shall be applied. If this flag is set to Disable (0), the motion tracker's **Z**-axis positioning shall be ignored.
Roll Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the roll (**X**-axis rotation) of the motion tracker. If this flag is set to Enable (1), the motion tracker's roll rotation shall be applied. If this flag is set to Disable (0), the motion tracker's roll rotation shall be ignored.
Pitch Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the pitch (**Y**-axis rotation) of the motion tracker. If this flag is set to Enable (1), the motion tracker's pitch rotation shall be applied. If this flag is set to Disable (0), the motion tracker's pitch rotation shall be ignored.
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	The value of this parameter is used to enable or disable the yaw (**Z**-axis rotation) of the motion tracker. If this flag is set to Enable (1), the motion tracker's yaw rotation shall be applied. If this flag is set to Disable (0), the motion tracker's yaw rotation shall be ignored.

Parameter	Description
View/View Group Select Type: 1-bit field Units: N/A Values: 0 View 1 View Group Default: IG-configurable	The value of this parameter specifies whether the tracking device is attached to a single view or a view group. If set to View (0), the *View/View Group ID* parameter shall identify a single view. If set to View Group (1), that parameter shall identify a view group.
View/View Group ID Type: unsigned int16 Units: N/A Default: 0	The value of this parameter specifies the view or view group to which the tracking device shall be attached.

6.1.19 Earth Reference Model Definition

The default ERM used for geodetic positioning shall be WGS 84. The Host may define another ERM by sending an **Earth Reference Model Definition** packet to the IG. This packet defines the equatorial radius and the flattening of the new reference ellipsoid.

When the IG receives an **Earth Reference Model Definition** packet, it shall set the *Earth Reference Model* parameter of the **Start of Frame** packet appropriately. If, for some reason, the IG is not able to support the ERM defined by the Host, the parameter shall be set to WGS 84 (0).

The contents of the **Earth Reference Model Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = 012h															
Reserved								*1	Reserved																						
Equatorial Radius																															
Flattening																															

*1 *Custom ERM Enable*

Figure 77 – Earth Reference Model Definition Packet Structure

Table 25 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 25 – Earth Reference Model Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 012h	This parameter identifies this data packet as the **Earth Reference Model Definition** packet. The Host shall set this parameter to 012h.

Parameter	Description
Custom ERM Enable Type: 1-bit field Units: N/A Values: 0 Disable (use WGS 84) 1 Enable Default: 0	The value of this parameter specifies whether the IG may use the Earth Reference Model (ERM) defined by this packet. If this parameter is set to Disable (0), the IG shall use the WGS 84 reference model and all other parameters in this packet shall be ignored. If this parameter is set to Enable (1), the IG shall use the *Equatorial radius* and *Flattening* values to characterize the ellipsoid.
Equatorial Radius Type: double float Units: meters Default: 6,378,137.0	The value of this parameter shall be applied as the semi-major axis of the ellipsoid.
Flattening Type: double float Units: meters Default: $\frac{1}{298.257223563}$	The value of this parameter shall be applied as the flattening of the ellipsoid. This value shall be calculated as follows: $$f = \frac{(a-b)}{a}$$ where f is the flattening, a is the semi-major axis (equatorial radius), and b is the semi-minor axis (polar radius). A flattening value of 0.0 defines a spherical Earth.

6.1.20 Acceleration Control

The **Acceleration Control** packet is used to define linear and angular accelerations for entities and articulated parts. This packet is commonly used in conjunction with the **Velocity Control** packet (Section 6.1.8).

Accelerations may also be used to enable the IG to compensate for transport delays or jitter produced by asynchronous operation or in dead-reckoning computations in synchronous operation. An **Acceleration Control** packet may be sent each frame in conjunction with **Entity Position** (6.1.2) and **Velocity Control** (6.1.8) packets or alone for an active entity that has been positioned previously with an **Entity Position** packet. This provides the IG with enough information to perform second order extrapolation to compute the entity's probable position during the next frame if smoothing has been enabled for the entity in the **Entity Control** packet (6.1.38).

When acceleration is specified for an entity or articulated part, the IG shall maintain that acceleration until a new acceleration is specified by the Host. If the Host changes the position and/or orientation of an entity or articulated part, the IG shall perform the transformation and extrapolation shall continue from that state beginning with the next frame. If the Host sets all acceleration components to zero, acceleration shall no longer be used in any extrapolation or dead-reckoning algorithms for the entity or articulated part.

The contents of the **Acceleration Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 013h															
Entity ID																*Articulated Part ID*								Reserved						*2	*1
X Linear Acceleration																															
Y Linear Acceleration																															
Z Linear Acceleration																															
Roll Angular Acceleration																															
Pitch Angular Acceleration																															
Yaw Angular Acceleration																															

*1 *Apply to Articulated Part*
*2 *Coordinate System*

Figure 78 – Acceleration Control Packet Structure

Table 26 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 26 – Acceleration Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 013h	This parameter identifies this data packet as the **Acceleration Control** packet. The Host shall set this parameter to 013h.
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the packet data shall be applied.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter specifies the articulated part to which the acceleration may be applied. If the *Apply to Articulated Part* flag is set to True (1), this parameter shall identify the articulated part belonging to the entity specified by *Entity ID*. If the flag is set to False (0), the IG shall ignore this parameter.
Apply to Articulated Part Type: 1-bit field Units: N/A Values: 0 False 1 True	This parameter specifies whether the acceleration may be applied to an articulated part or an entity. If this flag is set to True (1), the acceleration shall be applied to the articulated part specified by the *Articulated Part ID*. If this flag is set to False (0), the acceleration shall be applied to the entity specified by the *Entity ID* parameter.

Parameter	Description
Coordinate System Type: 1-bit field Units: N/A Values: 0 World/Parent 1 Local	This parameter specifies the reference coordinate system to which the linear and angular accelerations may be applied. When this parameter is set to World/Parent (0) and the entity is a top-level (non-child) entity, the accelerations shall be defined relative to the Geodetic Reference Plane with NED Coordinate System as described in Section 5.4.1.2. Linear accelerations shall describe a path along and above the surface of the geoid and angular accelerations shall describe a rotation relative to the axes of the reference plane as shown in Figure 18. When this parameter is set to World/Parent (0) and the entity is a child entity, the accelerations shall be defined relative to the parent's local coordinate system as described in Section 5.4.2.2. When this parameter is set to Local (1), the accelerations shall be defined relative to the entity's local coordinate system. This parameter shall be ignored if *Apply to Articulated Part* is set to True (1).
X Linear Acceleration Type: single float Units: m/s^2 Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the X component of a linear acceleration vector. The IG shall continually move the specified entity or articulated part along this linear acceleration vector.
Y Linear Acceleration Type: single float Units: m/s^2 Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the Y component of a linear acceleration vector. The IG shall continually move the specified entity or articulated part along this linear acceleration vector.

Parameter	Description
Z Linear Acceleration Type: single float Units: m/s^2 Default: 0 Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the Z component of a linear acceleration vector. The IG shall continually move the specified entity or articulated part along this linear acceleration vector.
Roll Angular Acceleration Type: single float Units: deg/s^2 Default: Model-dependent Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **X** axis after yaw and pitch have been applied. The IG shall rotate the specified entity or articulated part at this angular acceleration and shall honor the order of rotation described in Section 5.4.
Pitch Angular Acceleration Type: single float Units: deg/s^2 Default: Model-dependent Datum: Entities: As specified by *Coordinate System* parameter Articulated Parts: Submodel coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **Y** axis after yaw and pitch have been applied. The IG shall rotate the specified entity or articulated part at this angular acceleration and shall honor the order of rotation described in Section 5.4.

Parameter			Description
Yaw Angular Acceleration			This parameter specifies the angle of rotation of the articulated part submodel about its **Z** axis after yaw and pitch have been applied. The IG shall rotate the specified entity or articulated part at this angular acceleration and shall honor the order of rotation described in Section 5.4.
Type:	single float		
Units:	deg/s^2		
Default:	Model-dependent		
Datum:	Entities:	As specified by *Coordinate System* parameter	
	Articulated Parts:	Submodel coordinate system	

6.1.21 View Definition

The **View Definition** packet allows the Host to override the IG's default configuration for a view. This packet is used to specify the projection type, to define the size of the viewing volume, and to assign the view to a view group. Subsequent **View Definition** packets specifying the same view shall redefine the shape and behavior of that view. If more than one **View Definition** packet specifying the same view is received in the same frame, the last one in the message shall be used.

Refer to Section 5.2 for details on projection types, viewing volumes, and view groups.

The contents of the **View Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 40																*Packet ID* = 014h															
View ID																*Group ID*								*7	*6	*5	*4	*3	*2	*1	
View Type				*10	*9	*8		Reserved																							
Near																															
Far																															
Left																															
Right																															
Top																															
Bottom																															
Reserved																															

*1 *Near Enable*
*2 *Far Enable*
*3 *Left Enable*
*4 *Right Enable*
*5 *Top Enable*
*6 *Bottom Enable*
*7 *Mirror Mode*
*8 *Pixel Replication Mode*
*9 *Projection Type*
*10 *Reorder*

Figure 79 – View Definition Packet Structure

Table 27 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 27 – View Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 40.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 014h	This parameter identifies this data packet as the **View Definition** packet. The Host shall set this parameter to 014h.
View ID Type: unsigned int16 Units: N/A	The value of this parameter shall determine the view to which the data in this packet is applied.
Group ID Type: unsigned int8 Units: N/A Values: 0 None 1 – 255 Specifies view group	The value of this parameter shall determine the group to which the view is to be assigned. If this value is zero (0), the view shall not be assigned to a group.
Near Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the near clipping plane shall be set to the value of the *Near* parameter within this packet. If this parameter is set to Disable (0), the *Near* parameter shall be ignored.
Far Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the far clipping plane shall be set to the value of the *Far* parameter within this packet. If this parameter is set to Disable (0), the *Far* parameter shall be ignored.
Left Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the left half-angle of the view frustum shall be set according to the value of the *Left* parameter within this packet. If this parameter is set to Disable (0), the *Left* parameter shall be ignored.

Parameter	Description
Right Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the right half-angle of the view frustum shall be set according to the value of the *Right* parameter within this packet. If this parameter is set to Disable (0), the *Right* parameter shall be ignored.
Top Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the top half-angle of the view frustum shall be set according to the value of the *Top* parameter within this packet. If this parameter is set to Disable (0), the *Top* parameter shall be ignored.
Bottom Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	The value of this parameter specifies whether the bottom half-angle of the view frustum shall be set according to the value of the *Bottom* parameter within this packet. If this parameter is set to Disable (0), the *Bottom* parameter shall be ignored.
Mirror Mode Type: unsigned 2-bit field Units: N/A Values: 0 None 1 Horizontal 2 Vertical 3 Horizontal and Vertical Default: IG-configurable	The value of this parameter shall determine the mirroring function to be performed on the view. This feature is typically used to replicate the view of a mirrored surface such as a rear view mirror.
Pixel Replication Mode Type: unsigned 3-bit field Units: N/A Values: 0 None 1 1×2 2 2×1 3 2×2 4 – 7 Defined by IG Default: IG-configurable	The value of this parameter shall determine the pixel replication function to be performed on the view. This feature is typically used in sensor applications to perform electronic zooming (i.e.,, pixel and line doubling).

Parameter	Description
Projection Type Type: 1-bit field Units: N/A Values: 0 Perspective 1 Orthographic Parallel Default: IG-configurable	The value of this parameter shall determine whether the view projection may be perspective (Section 5.2.1.1) or orthographic parallel (Section 5.2.1.2).
Reorder Type: 1-bit field Units: N/A Values: 0 No Reorder 1 Bring to Top Default: IG-configurable	The value of this parameter shall determine whether the view may be moved to the top of any overlapping views. In cases where multiple overlapping views are moved to the top, the last view specified shall have priority.
View Type Type: unsigned 3-bit field Units: N/A Values: 0 – 7 Default: IG-configurable	The value of this parameter specifies an IG-defined type for the indicated view. For example, a Host might switch a view type from out-the-window to IR for a given channel. If the view type specified is not supported, the IG shall gracefully ignore this packet.
Near Type: single float Units: meters Values: > 0 to < *Far* Default: IG-configurable	The value of this parameter shall be applied as the position of the view's near clipping plane. This distance shall be measured along the viewing vector from the eyepoint to the plane.
Far Type: single float Units: meters Values: > *Near* Default: IG-configurable	The value of this parameter shall be applied as the position of the view's far clipping plane. This distance shall be measured along the viewing vector from the eyepoint to the plane.

Parameter	Description
Left Type: single float Units: degrees Values: > -90.0 to < *Right* Default: IG-configurable	The value of this parameter shall be applied as the left half-angle of the view frustum. This value shall be the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 40).
Right Type: single float Units: degrees Values: > *Left* to < 90.0 Default: IG-configurable	The value of this parameter shall be applied as the right half-angle of the view frustum. This value shall be the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 40).
Top Type: single float Units: degrees Values: > *Bottom* to < 90.0 Default: IG-configurable	The value of this parameter shall be applied as the top half-angle of the view frustum. This value shall be the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 40).
Bottom Type: single float Units: degrees Values: > -90.0 to < *Top* Default: IG-configurable	The value of this parameter shall be applied as the bottom half-angle of the view frustum. This value shall be the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 40).

6.1.22 Collision Detection Segment Definition

The **Collision Detection Segment Definition** packet enables the Host to define one or more collision detection segments for an entity. A collision detection segment is a line segment against which collision testing is performed by the IG. When a collision detection segment intersects a polygon, the IG shall register a collision by sending a **Collision Detection Segment Notification** (Section 6.2.13) packet to the Host identifying the segment and the object with which it collided.

Note that collision detection testing shall be performed every frame by the IG.

The segment is defined by specifying the locations of its endpoints with respect to the associated entity's body coordinate system. Figure 80 illustrates five segments defined for an aircraft:

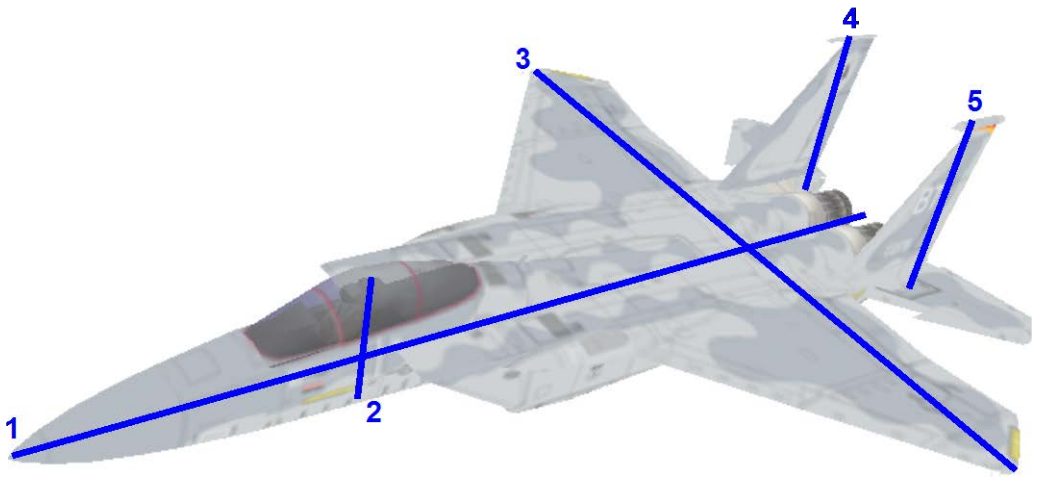


Figure 80 – Examples of Collision Detection Segments

Collision detection segments shall only be tested segment-to-polygon. Collision detection shall not be performed between an entity's segments and its own geometry.

If the *Collision Reporting Enable* parameter of an **Entity Control** packet is set to Disabled (0), the referenced entity's segments shall not be used for collision detection segment testing. If the state of an entity is set to Inactive/Standby (0) via the *Entity State* parameter of an **Entity Control** packet, neither that entity's segments nor its geometry shall be included in collision detection segment testing.

If an entity is destroyed, any collision detection segments defined for that entity shall also be destroyed.

The Host shall only create collision detection segments by referencing an entity. If a segment needs to be defined along a non-entity object, the Host shall first create an entity with no geometry (entity type zero) to represent that object.

Since collision tests are conducted at discrete moments in time, it is possible that a segment could pass completely through a polygon between successive tests, causing a missed collision. It may therefore be necessary for the IG to use segment sweeping or some other mechanism to avoid this situation.

The contents of the **Collision Detection Segment Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 40																*Packet ID* = 015h															
Segment ID								Reserved								*1								*Entity ID*							
X1																															
Y1																															
Z1																															
X2																															
Y2																															
Z2																															
Material Mask																															
Reserved																															

*1 *Segment Enable*

Figure 81 – Collision Detection Segment Definition Packet Structure

Table 28 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 28 – Collision Detection Segment Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 40.
Packet ID Type: unsigned int16 Units: N/A Value: 015h	This parameter identifies this data packet as the **Collision Detection Segment Definition** packet. The Host shall set this parameter to 015h.
Segment ID Type: unsigned int8 Units: N/A	The value of this parameter shall determine the ID of the segment. If an ID is specified for which a segment is already defined, that segment shall be overwritten.
Segment Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter specifies whether the segment is enabled or disabled. If this parameter is set to Disable (0), the specified segment shall be ignored during collision testing. If this parameter is set to Enable (1), the specified segment shall be considered during collision testing.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the packet data shall be applied.
X1 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the X offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The X offset of the other endpoint is defined by the *X2* parameter.
Y1 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Y offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The Y offset of the other endpoint is defined by the *Y2* parameter.
Z1 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Z offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The Z offset of the other endpoint is defined by the *Z2* parameter.
X2 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the X offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The X offset of the other endpoint is defined by the *X1* parameter.
Y2 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Y offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The Y offset of the other endpoint is defined by the *Y1* parameter.

Parameter	Description
Z2 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Z offset of one endpoint of the collision segment. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter. The Z offset of the other endpoint is defined by the *Z1* parameter.
Material Mask Type: unsigned int32 Units: N/A Default: IG-configurable	The value of this parameter shall determine the environmental and cultural features to be included in or excluded from consideration for collision testing. Each bit represents a range of material code values. Setting that bit to one (1) shall cause the IG to register hits with materials within the corresponding range. Setting this field to 0h has the effect of disabling collision testing for this segment. Refer to the appropriate IG documentation for material code assignments.

6.1.23 Collision Detection Volume Definition

The **Collision Detection Volume Definition** packet enables the Host to define one or more collision detection volumes for an entity. A collision detection volume is a sphere or a cuboid through which collision testing is performed by the IG. When a collision detection volume passes through another collision detection volume, the IG shall register a collision by sending a **Collision Detection Volume Notification** (Section 6.2.14) packet to the Host identifying the collided volumes.

Note that collision detection testing shall be performed every frame by the IG.

The Host defines a volume by specifying its location, size, and orientation with respect to the associated entity's body coordinate system. A sphere's size is specified as a radius; a cuboid's size is specified by its width, height, and depth. Figure 82 illustrates two cuboid volumes defined for an aircraft:

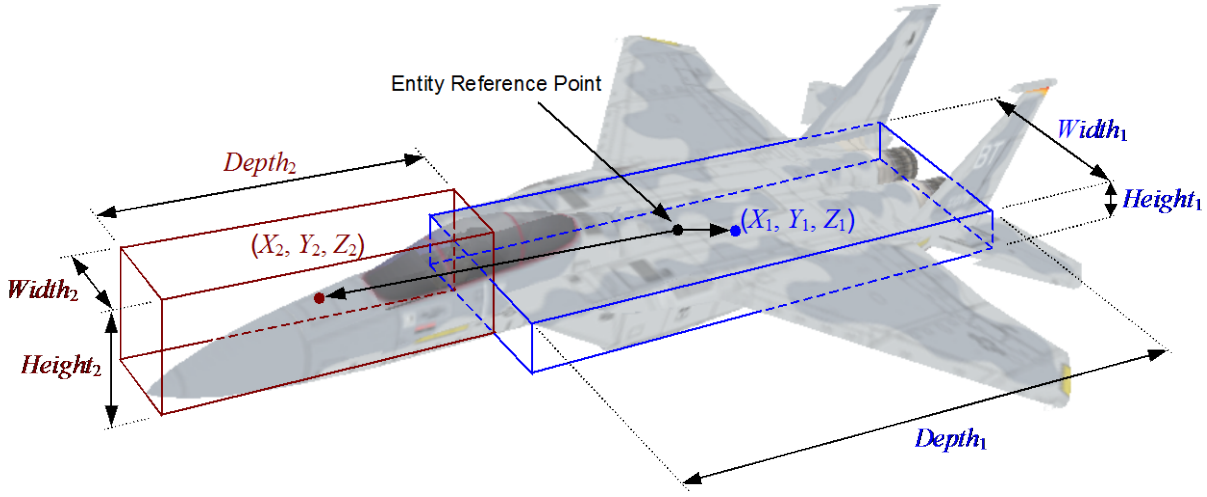


Figure 82 – Examples of Collision Detection Volumes

Unlike collision detection segments, which are tested segment-to-polygon, collision detection volumes shall be tested volume-to-volume. Volumes shall not be tested against polygons or collision segments. Volumes associated with the same entity shall not be tested against each other.

Since collision tests are conducted at discrete moments in time, it is possible that two volumes could pass completely through one another between successive tests, causing a missed collision. It may therefore be necessary for the IG to use volume sweeping or some other mechanism to avoid this situation.

If the state of an entity is set to Inactive/Standby (0) via the *Entity State* parameter of an **Entity Control** packet, collision detection volume testing shall not be performed for that entity.

If the *Collision Reporting Enable* parameter of the **Entity Control** packet is set to Disabled (0), volumes defined for the entity shall not be used as "source" volumes for collision testing. Figure 83 below illustrates the individual tests that would occur each frame between a hypothetical group of entities:

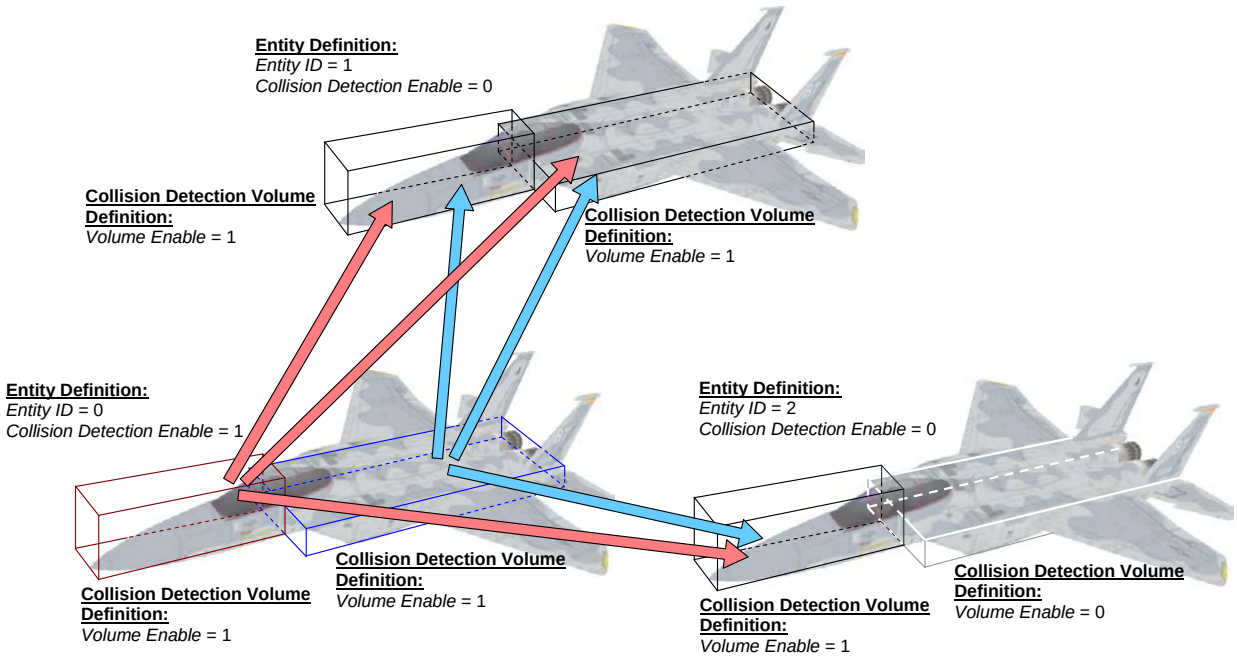


Figure 83 – Collision Volume Testing Between Multiple Entities

The illustration shows three aircraft with two collision detection volumes defined for each. The *Collision Reporting Enable* parameter has been set to Enabled (1) for Entity 0 and to Disabled (0) for Entities 1 and 2. This means that only the volumes associated with Entity 0 shall be used as the sources of collision testing. The two source volumes are tested with every other enabled volume not associated with Entity 0. Note that one of the volumes defined for Entity 2 is disabled; that volume is not included in *any* collision testing.

If collision detection is enabled for two intersecting entities, two collisions shall be reported for each pair of intersecting volumes.

If an entity is destroyed, any collision detection volumes defined for that entity shall also be destroyed.

The Host shall only create collision detection volumes by referencing an entity. If a volume needs to be defined about a non-entity object, the Host shall first create an entity with no geometry (entity type zero) to represent that object.

The contents of the **Collision Detection Volume Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 48																*Packet ID* = 016h															
Volume ID								Reserved						*2	*1	*Entity ID*															
X																															
Y																															
Z																															
Height/Radius																															
Width																															
Depth																															
Roll																															
Pitch																															
Yaw																															
Reserved																															

*1 *Volume Enable*
*2 *Volume Type*

Figure 84 – Collision Detection Volume Definition Packet Structure

Table 29 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 29 – Collision Detection Volume Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 48.
Packet ID Type: unsigned int16 Units: N/A Value: 016h	This parameter identifies this data packet as the **Collision Detection Volume Definition** packet. The Host shall set this parameter to 016h.
Volume ID Type: unsigned int8 Units: N/A	The value of this parameter shall determine the ID of the volume. If an ID is specified for which a volume is already defined, that volume shall be overwritten.

Parameter	Description
Volume Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	The value of this parameter specifies whether the volume is enabled or disabled. If this parameter is set to Disable (0), the specified volume shall be ignored during collision testing. If this parameter is set to Enable (1), the specified volume shall be considered during collision testing.
Volume Type Type: 1-bit field Units: N/A Values: 0 Sphere 1 Cuboid Default: IG-configurable	The value of this parameter shall determine whether the volume is a sphere or cuboid.
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the packet data shall be applied.
X Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the X offset of the center of the volume. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter.
Y Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Y offset of the center of the volume. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter.
Z Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	The value of this parameter shall be applied as the Z offset of the center of the volume. This offset shall be measured with respect to the coordinate system of the entity specified by the *Entity ID* parameter.

Parameter	Description
Radius (Spherical Volumes) Type: single float Units: meters Values: > 0 Default: IG-configurable	For spherical collision detection volumes, the value of this parameter shall be applied as the radius of the sphere.
Height (Cuboid Volumes) Type: single float Units: meters Values: > 0 Default: IG-configurable	For cuboid collision detection volumes, the value of this parameter shall be applied as the length of the cuboid along its **Z** axis.
Width Type: single float Units: meters Values: > 0 Default: IG-configurable	For cuboid collision detection volumes, the value of this parameter shall be applied as the length of the cuboid along its **Y** axis. This parameter shall be ignored if *Volume Type* is set to Sphere (0).
Depth Type: single float Units: meters Values: > 0 Default: IG-configurable	For cuboid collision detection volumes, the value of this parameter shall be applied as the length of the cuboid along its **X** axis. This parameter shall be ignored if *Volume Type* is set to Sphere (0).
Roll Type: single float Units: degrees Values: -180 – 180 Default: IG-configurable Datum: Entity reference plane	For cuboid collision detection volumes, the value of this parameter shall be applied as the roll of the cuboid with respect to the entity's coordinate system. This parameter may be ignored if *Volume Type* is set to Sphere (0).

Parameter	Description
Pitch Type: single float Units: degrees Values: -90 – 90 Default: IG-configurable Datum: Entity reference plane	For cuboid collision detection volumes, the value of this parameter shall be applied as the pitch of the cuboid with respect to the entity's coordinate system. This parameter may be ignored if *Volume Type* is set to Sphere (0).
Yaw Type: single float Units: degrees Values: 0 – 360 Default: IG-configurable Datum: Entity reference coordinate system	For cuboid collision detection volumes, the value of this parameter shall be applied as the yaw of the cuboid with respect to the entity's coordinate system. This parameter may be ignored if *Volume Type* is set to Sphere (0).

6.1.24 HAT/HOT Request

The **HAT/HOT Request** packet is used by the Host to request the Height Above Terrain (HAT) of a specified point and/or the Height Of Terrain (HOT) below a specified test point. The test point may be defined with respect to either the Geodetic coordinate system or an entity's body coordinate system.

Each request is identified by the *HAT/HOT ID* parameter. When the IG responds to the request, it shall set the HAT/HOT *ID* parameter of the response packet to match that in the request.

The *Update Period* parameter specifies the number of frames between periodic responses. This allows the Host to send just one **HAT/HOT Request** packet but receive continuous responses if the test point does not move with respect to the specified coordinate system. If *Update Period* is set to zero, the request shall be treated as a one-shot request and the IG shall return a single response. The Host should manipulate the value of *HAT/HOT ID* so that an ID is not reused before the IG has sufficient time to process and respond to the request [CIGIBP, Request IDs]. If *Update Period* is set to some value n greater than zero, the IG shall return a request every n^{th} frame until the Entity is destroyed or until the *Update Period* parameter is set to zero.

If the *Request Type* parameter is set to HAT (0) or HOT (1), the IG shall respond with a **HAT/HOT Response** packet (Section 6.2.2) containing the requested datum. If the parameter is set to Extended HAT/HOT (2), the IG shall respond with a **HAT/HOT Extended Response** packet (Section 6.2.3) containing *both* data, along with the surface material code and normal vector.

The IG can only return valid HAT and/or HOT data if the test point is located within the bounds of the current database. If the HAT or HOT cannot be calculated, the *Valid* parameter of the response packet shall be set to Invalid (0).

Besides the range of the *HAT/HOT ID* parameter, there is no restriction on the number of HAT and/or HOT requests that may be sent in a single frame, however, the response time of the IG might be degraded as the number of requests increases.

The contents of the **HAT/HOT Request** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 40</td><td colspan="16">Packet ID = 017h</td></tr>
<tr><td colspan="16">HAT/HOT ID</td><td colspan="16">Entity ID</td></tr>
<tr><td colspan="13">Reserved</td><td>*2</td><td>*1</td><td colspan="14">Update Period</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="32">Reserved</td></tr>
<tr><td colspan="32">Latitude/X Offset</td></tr>
<tr><td colspan="32"></td></tr>
<tr><td colspan="32">Longitude/Y Offset</td></tr>
<tr><td colspan="32"></td></tr>
<tr><td colspan="32">Altitude/Z Offset</td></tr>
<tr><td colspan="32"></td></tr>
</table>

*1 *Request Type*
*2 *Coordinate System*

Figure 85 – HAT/HOT Request Packet Structure

Table 30 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 30 – HAT/HOT Request Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 40.
Packet ID Type: unsigned int16 Units: N/A Value: 017h	This parameter identifies this data packet as the **HAT/HOT Request** packet. The Host shall set this parameter to 017h.
HAT/HOT ID Type: unsigned int16 Units: N/A	The value of this parameter shall identify the HAT/HOT request [CIGIBP, Request IDs]. When the IG returns a **HAT/HOT Response** or **HAT/HOT Extended Response** packet in response to this request, the *HAT/HOT ID* parameter of that packet shall contain this value to correlate the response with this request.
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the packet data shall be relative. This parameter shall be ignored if **Coordinate System** is set to Geodetic (0).
Request Type Type: unsigned 2-bit field Units: N/A Values: 0 HAT 1 HOT 2 Extended	The value of this parameter determines what type of response packet the IG may return for this request. If this parameter is set to HAT (0), the IG shall respond with a **HAT/HOT Response** packet containing the Height Above Terrain. If this parameter is set to HOT (1), the IG shall respond with a **HAT/HOT Response** packet containing the Height Of Terrain. If this parameter is set to Extended (2), the IG shall respond with a **HAT/HOT Extended Response** packet, which contains both the Height Above Terrain and the Height Of Terrain.

Parameter	Description
Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter specifies the coordinate system within which the test point is defined. If this parameter is set to Geodetic (0), the test point shall be defined as a Latitude, Longitude, and Altitude. If this parameter is set to Entity (1), the test point shall be defined as X, Y, and Z offsets from the reference point of the entity specified by *Entity ID*.
Update Period Type: unsigned int8 Units: N/A Values: 0 One-Shot request > 0 Indicates update period	The value of this parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG shall return a single response. A value of $n > 0$ indicates that the IG shall return a response every n^{th} frame.
Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	If *Coordinate System* is set to Geodetic (0), this parameter shall indicate the latitude of the HAT/HOT request.
X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	If *Coordinate System* is set to Entity (1), this parameter shall be applied as the X offset of the point of the HAT/HOT request.

Parameter	Description
Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	If *Coordinate System* is set to Geodetic (0), this parameter shall indicate the longitude of the HAT/HOT request.
Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	If *Coordinate System* is set to Entity (1), this parameter shall be applied as the Y offset of the point of the HAT/HOT request.
Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If *Coordinate System* is set to Geodetic (0), this parameter shall indicate the altitude of the HAT/HOT request. This parameter shall be ignored if *Request Type* is set to HOT (1).
Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	If *Coordinate System* is set to Entity (1), this parameter shall be applied as the Z offset of the point of the HAT/HOT request.

6.1.25 Line of Sight Segment Request

Line-of-Sight (LOS) Segment testing is used to determine whether an object lies along a test segment. This type of test is typically used to determine whether one point is visible from another, or whether the point is occluded by some object. The Line of Sight test segment is defined in the **Line of Sight Segment Request** packet by a source point and a destination point.

The *LOS ID* parameter is used to correlate requests from the Host with responses from the IG. When the IG responds to a LOS request, it shall copy the *LOS ID* value contained within the request to the *LOS ID* parameter of the corresponding response packet.

Note that **Line of Sight Segment Request** packets and **Line of Sight Vector Request** packets share the *LOS ID* parameter. Therefore the *LOS ID* value should not be duplicated between the **Line of Sight Segment Request** packets and **Line of Sight Vector Request** packets. Further the Host should manipulate the value of *LOS ID* so that the ID is not reused before the IG has sufficient time to respond to the LOS request [CIGIBP, Request IDs]. These actions will prevent similarly identified requests from being lost by the IG.

If the *Request Type* parameter is set to Basic (0), the IG shall respond with a **Line of Sight Response** packet (Section 6.2.4). If the parameter is set to Extended (1), the IG shall respond with a **Line of Sight Extended Response** packet (Section 6.2.5).

The *Alpha Threshold* parameter specifies the minimum alpha value with which an intersection shall be reported. If an LOS test segment intersects with a surface whose alpha at the intersection point is lower than this value, no **Line of Sight Response** or **Line of Sight Extended Response** packet shall be generated.

The *Update Period* parameter specifies the number of frames between periodic responses. This allows the Host to send just one **Line of Sight Segment Request** packet but receive continuous responses if the test point does not move with respect to the specified coordinate system. If *Update Period* is set to zero, the request shall be treated as a one-shot request and the IG shall return a single response. If *Update Period* is set to some value n greater than zero, the IG shall return a request every n^{th} frame until the Entity is destroyed or until the *Update Period* parameter set to zero.

The IG can only return valid LOS data if an intersection is detected along the LOS segment. If the LOS data cannot be calculated, the *Valid* parameter of the response packet shall be set to zero (0).

The IG shall generate a response for each intersection along the LOS segment.

Besides the range of the *LOS ID* parameter, there is no restriction on the number of LOS requests that may be sent in a single frame; however, the response time of the IG might be degraded as the number of LOS requests increases.

The contents of the **Line of Sight Segment Request** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 72																*Packet ID* = 018h															
LOS ID																*Source Entity ID*															
Reserved			*5	*4	*3	*2	*1	*Alpha Threshold*								*Destination Entity ID*															
Update Period								Reserved																							
Source Latitude/X Offset																															
Source Longitude/Y Offset																															
Source Altitude/Z Offset																															
Destination Latitude/X Offset																															
Destination Longitude/Y Offset																															
Destination Altitude/Z Offset																															
Material Mask																															
Reserved																															

*1 *Request Type*
*2 *Source Point Coordinate System*
*3 *Destination Point Coordinate System*
*4 *Response Coordinate System*
*5 *Destination Entity ID Valid*

Figure 86 – Line of Sight Segment Request Packet Structure

Table 31 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 31 – Line of Sight Segment Request Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 72	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 72.
Packet ID Type: unsigned int16 Units: N/A Value: 018h	This parameter identifies this data packet as the **Line of Sight Segment Request** packet. The Host shall set this parameter to 018h.

Parameter	Description
LOS ID Type: unsigned int16 Units: N/A	The value of this parameter identifies the LOS request [CIGIBP, Request IDs]. When the IG returns a **Line of Sight Response** packet in response to this request, the *LOS ID* parameter of that packet shall contain this value to correlate the response with this request. Note: Because the **Line of Sight Response** data packet is used for responding to both the LOS segment and LOS vector requests, the *LOS ID* value used for one request type should not be duplicated for the other request type before the IG has sufficient time to generate a response.
Source Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the test segment endpoints shall be relative. This parameter shall be ignored if *Source Point Coordinate System* and *Destination Point Coordinate System* are both set to Geodetic (0).
Request Type Type: 1-bit field Units: N/A Values: 0 Basic 1 Extended	The value of this parameter determines what type of response the IG may return for this request. If this parameter is set to Basic (0), the IG shall respond with a **Line of Sight Response** packet. If this parameter is set to Extended (1), the IG shall respond with a **Line of Sight Extended Response** packet.
Source Point Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter indicates the coordinate system relative to which the test segment source endpoint is specified. If this parameter is set to Geodetic (0), then the endpoint shall be given by latitude, longitude, and altitude. If this parameter is set to Entity (1), then the endpoint shall be defined relative to the reference point of the entity specified by *Entity ID*.

Parameter	Description
Destination Point Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter indicates the coordinate system relative to which the test segment destination endpoint is specified. If this parameter is set to Geodetic (0), then the endpoint shall be given by latitude, longitude, and altitude. If this parameter is set to Entity (1) and *Destination Entity ID Valid* is set to Not Valid (0), then the endpoint shall be defined relative to the reference point of the entity specified by *Source Entity ID*. If this parameter is set to Entity (1) and *Destination Entity ID Valid* is set to Valid (1), then the endpoint shall be defined relative to the reference point of the entity specified by *Destination Entity ID*.
Response Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter specifies the coordinate system to be used in the response. If this parameter is set to Geodetic (0), then the intersection point shall be specified by latitude, longitude, and altitude. If this parameter is set to Entity (1), then the intersection point shall be specified relative to the reference point of the intersected entity.
Destination Entity ID Valid Type: unsigned int16 Units: N/A Values: 0 Not Valid 1 Valid	The value of this parameter determines whether the *Destination Entity ID* parameter contains a valid entity ID. If this flag is set to Valid (1) and *Destination Point Coordinate System* is set to Entity (1), then the destination endpoint shall be defined with respect to the entity specified by *Destination Entity ID*. If this flag is set to Not Valid (0), then the destination endpoint shall be defined with respect to either the source entity (specified by *Source Entity ID*) or the Geodetic coordinate system as determined by the *Destination Point Coordinate System* parameter.
Alpha Threshold Type: unsigned int8 Units: N/A	The value of this parameter shall define the minimum alpha value (i.e.,, minimum opacity) a surface may have for an LOS response to be generated.

Parameter	Description
Destination Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the *Destination X Offset*, *Destination Y Offset*, and *Destination Z Offset* parameters shall apply. This parameter shall be used only if the *Destination Point Coordinate System* parameter is set to Entity (1) and the *Destination Entity ID Valid* flag is set to Valid (1).
Update Period Type: unsigned int8 Units: N/A Values: 0 One-Shot request > 0 Indicates update period	The value of this parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG shall return a single response. A value of $n > 0$ indicates that the IG shall return a response every n^{th} frame.
Source Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	If *Source Point Coordinate System* is set to Geodetic (0), this parameter shall indicate the latitude of the source endpoint of the LOS test segment.
Source X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the X offset of the source endpoint of the LOS test segment.
Source Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	If *Source Point Coordinate System* is set to Geodetic (0), this parameter shall indicate the longitude of the source endpoint of the LOS test segment.
Source Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Y offset of the source endpoint of the LOS test segment.

Parameter	Description
Source Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If *Source Point Coordinate System* is set to Geodetic (0), this parameter shall be applied as the altitude of the source endpoint of the LOS test segment.
Source Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Z offset of the source endpoint of the LOS test segment.
Destination Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	If *Destination Point Coordinate System* is set to Geodetic (0), this parameter shall indicate the latitude of the destination endpoint of the LOS test segment.
Destination X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Destination Point Coordinate System* is set to Entity (1), this parameter shall be applied as the X offset of the destination endpoint of the LOS test segment. This offset may be relative to either the source entity or destination entity, depending upon the value of the *Destination Entity ID Valid* flag.
Destination Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	If *Destination Point Coordinate System* is set to Geodetic (0), this parameter shall indicate the longitude of the destination endpoint of the LOS test segment.
Destination Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Destination Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Y offset of the destination endpoint of the LOS test segment. This offset may be relative to either the source entity or destination entity, depending upon the value of the *Destination Entity ID Valid* flag.

Parameter	Description
Destination Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If *Destination Point Coordinate System* is set to Geodetic (0), this parameter shall be applied as the altitude of the destination endpoint of the LOS test segment.
Destination Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Destination Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Z offset of the destination endpoint of the LOS test segment. This offset may be relative to either the source entity or destination entity, depending upon the value of the *Destination Entity ID Valid* flag.
Material Mask Type: unsigned int32 Units: N/A Default: IG-configurable	This parameter specifies the environmental and cultural features to be included in or excluded from consideration for LOS segment testing. Each bit represents a material code range; setting that bit to one (1) shall cause the IG to register intersections with polygons whose material codes are within that range. Material code ranges are IG-dependent. Refer to the appropriate IG documentation for material code assignments.

6.1.26 Line of Sight Vector Request

Line-of-Sight (LOS) Vector testing is used to determine the range from a source point to an object along a test vector. Applications may include but are not limited to laser range finding, determining range to target, and testing for weight on wheels. The Line of Sight test vector emanates from the source position specified in the **Line of Sight Vector Request** packet. A minimum and a maximum range are specified in order to constrain the search.

The *LOS ID* parameter is used to correlate requests from the Host with responses from the IG. When the IG responds to an LOS request, it shall copy the *LOS ID* value contained within the request to the *LOS ID* parameter of the corresponding response packet.

Note that **Line of Sight Segment Request** packets and **Line of Sight Vector Request** packets share the *LOS ID* parameter. Therefore the *LOS ID* value should not be duplicated between the **Line of Sight Segment Request** packets and **Line of Sight Vector Request** packets. Further the Host should manipulate the value of *LOS ID* so that the ID is not reused before the IG has sufficient time to respond to the LOS request [CIGIBP, Request IDs].

If the *Request Type* parameter is set to Basic (0), the IG shall respond with a **Line of Sight Response** packet (Section 6.2.4). If the parameter is set to Extended (1), the IG shall respond with a **Line of Sight Extended Response** packet (Section 6.2.5).

The *Alpha Threshold* parameter specifies the minimum alpha value with which an intersection shall be reported. If an LOS test vector intersects with a surface whose alpha at the intersection point is lower than this value, no **Line of Sight Response** or **Line of Sight Extended Response** packet shall be generated.

The *Update Period* parameter specifies the number of frames between periodic responses. This allows the Host to send just one **Line of Sight Vector Request** packet but receive continuous responses if the test point does not move with respect to the specified coordinate system. If *Update Period* is set to zero, the request shall be treated as a one-shot request and the IG shall return a single response. If *Update Period* is set to some value n greater than zero, the IG shall return a request every n^{th} frame until the Entity is destroyed or until the *Update Period* parameter set to zero.

The IG can only return valid LOS data if an intersection is detected along the LOS segment, that is, between the minimum and maximum ranges specified. If the LOS data cannot be calculated, the *Valid* parameter of the response packet shall be set to zero (0).

The IG shall generate a response for each intersection along the LOS vector.

Besides the range of the *LOS ID* parameter, there is no restriction on the number of LOS requests that may be sent in a single frame; however, the response time of the IG might be degraded as the number of LOS requests increases.

The contents of the **Line of Sight Vector Request** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 64																*Packet ID* = 019h															
LOS ID																*Entity ID*															
Reserved					*3	*2	*1	*Alpha Threshold*								*Update Period*								Reserved							
Reserved																															
Azimuth																															
Elevation																															
Minimum Range																															
Maximum Range																															
Source Latitude/X Offset																															
Source Longitude/Y Offset																															
Source Altitude/Z Offset																															
Material Mask																															
Reserved																															

*1 *Request Type*
*2 *Source Point Coordinate System*
*3 *Response Coordinate System*

Figure 87 – Line of Sight Vector Request Packet Structure

Table 32 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 32 – Line of Sight Vector Request Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 64	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 64.
Packet ID Type: unsigned int16 Units: N/A Value: 019h	This parameter identifies this data packet as the **Line of Sight Vector Request** packet. The Host shall set this parameter to 019h.

Parameter	Description
LOS ID Type: unsigned int16 Units: N/A	The value of this parameter shall identify the LOS request [CIGIBP, Request IDs]. When the IG returns a **Line of Sight Response** packet in response to this request, the *LOS ID* parameter of that packet shall contain this value to correlate the response with this request. Note: Because the **Line of Sight Response** data packet is used for responding to both the LOS segment and LOS vector requests, the *LOS ID* value used for one request type should not be duplicated for the other request type before the IG has sufficient time to generate a response.
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity to which the test segment endpoints shall be relative. This parameter shall be ignored if *Source Point Coordinate System* is set to Geodetic (0).
Request Type Type: 1-bit field Units: N/A Values: 0 Basic 1 Extended	The value of this parameter determines what type of response the IG may return for this request. If this parameter is set to Basic (0), the IG shall respond with a **Line of Sight Response** packet. If this parameter is set to Extended (1), the IG shall respond with a **Line of Sight Extended Response** packet.
Source Point Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter indicates the coordinate system relative to which the test vector source point is specified. If this parameter is set to Geodetic (0), then the point shall be given by latitude, longitude, and altitude. The vector, specified by *Azimuth* and *Elevation*, shall be defined relative to the Geodetic coordinate system. If this parameter is set to Entity (1), then the point shall be defined relative to the reference point of the entity specified by *Entity ID*. The vector shall also be specified relative to the entity's coordinate system.

Parameter	Description
Response Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	The value of this parameter specifies the coordinate system to be used in the response. If this parameter is set to Geodetic (0), then the intersection point shall be reported as a latitude, longitude, and altitude. If this parameter is set to Entity (1), then the intersection point shall be reported as an XYZ offset relative to the reference point of the intersected entity.
Alpha Threshold Type: unsigned int8 Units: N/A	The value of this parameter shall define the minimum alpha value (i.e.,, minimum opacity) a surface may have for an LOS response to be generated.
Update Period Type: unsigned int8 Units: N/A Values: 0 One-Shot request > 0 Indicates update period	The value of this parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG shall return a single response. A value of $n > 0$ indicates that the IG shall return a response every n^{th} frame.
Azimuth Type: single float Units: degrees Values: -180.0 – 180.0 Datum: If *Source Point Coordinate System* = 0: True North If *Source Point Coordinate System* = 1: Entity's +**X** axis	The value of this parameter shall be applied as the horizontal angle of the LOS test vector.
Elevation Type: single float Units: degrees Values: -90.0 – 90.0 Datum: If *Source Point Coordinate System* = 0: Geodetic reference plane If *Source Point Coordinate System* = 1: Entity's **XY** plane	The value of this parameter shall be applied as the vertical angle of the LOS test vector. For vectors defined relative to the geodetic reference plane, a positive elevation shall produce a vector away from the ellipsoid. For vectors defined relative to an entity, a positive elevation shall produce an acute angle with the entity's –**Z** (up) axis.

Parameter	Description
Minimum Range Type: single float Units: meters Values: ≥ 0 Datum: LOS test vector source point	The value of this parameter specifies the minimum range along the LOS test vector at which intersection testing shall occur.
Maximum Range Type: single float Units: meters Values: > *Minimum Range* Datum: LOS test vector source point	The value of this parameter specifies the maximum range along the LOS test vector at which intersection testing shall occur.
Source Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90 – 90 Datum: Equator	If Source point Coordinate System is set to Geodetic (0), this parameter shall indicate the latitude of the source point of the LOS test vector.
Source X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the X offset of the source point of the LOS test vector.
Source Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180 – 180 Datum: Prime Meridian	If Source point Coordinate System is set to Geodetic (0), this parameter shall indicate the longitude of the source point of the LOS test vector.
Source Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Y offset of the source point of the LOS test vector.

Parameter	Description
Source Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If *Source Point Coordinate System* is set to Geodetic (0), this parameter shall be applied as the altitude of the source point of the LOS test vector.
Source Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If *Source Point Coordinate System* is set to Entity (1), this parameter shall be applied as the Z offset of the source point of the LOS test vector.
Material Mask Type: unsigned int32 Units: N/A	This parameter specifies the environmental and cultural features to be included in LOS segment testing. Each bit represents a material code range; setting that bit to one (1) shall cause the IG to register intersections with polygons whose material codes are within that range. Material code ranges are IG-dependent. Refer to the appropriate IG documentation for material code assignments.

6.1.27 Position Request

The **Position Request** packet is used to query the IG for the current position of an entity, articulated part, view, view group, or motion tracker. This feature may be useful for determining the locations of autonomous IG-driven entities, child entities and articulated parts, and view eyepoints. It may also be used for determining the instantaneous position and orientation of head trackers and other tracked input devices.

When the *Update Mode* parameter is set to Continuous (1), **Position Response** packets shall be sent to the Host each frame for valid objects and shall continue until the object is destroyed or a new **Position Request** for the same valid object is sent with the *Update Mode* set to One Shot (0). In the latter case, no further **Position Response** packets shall be sent for that object until a new request is made.

The contents of the **Position Request** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 8																*Packet ID* = 01Ah															
Articulated Part ID								*4				*3				*2				*1				*Object ID*							

*1 *Update Mode*
*2 *Object Class*
*3 *Coordinate System*
*4 Reserved

Figure 88 – Position Request Packet Structure

Table 33 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 33 – Position Request Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 8.
Packet ID Type: unsigned int16 Units: N/A Value: 01Ah	This parameter identifies this data packet as the **Position Request** packet. The Host shall set this parameter to 01Ah.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter identifies the articulated part whose position is being requested. The entity to which the part belongs shall be given by the *Object ID* parameter. This parameter shall be valid only when *Object Class* is set to Articulated Part (1).

Parameter	Description
Update Mode Type: 1-bit field Units: N/A Values: 0 One-Shot 1 Continuous	The value of this parameter specifies whether the IG may report the position of the requested object each frame. If this parameter is set to One-Shot (0), the IG shall report the position only one time. If this parameter is set to Continuous (1), the IG shall report the position each frame.
Object Class Type: unsigned 3-bit field Units: N/A Values: 0 Entity 1 Articulated Part 2 View 3 View Group 4 Motion Tracker	The value of this parameter shall indicate the type of object whose position is being requested.
Coordinate System Type: unsigned 2-bit field Units: N/A Values: 0 Geodetic 1 Parent Entity 2 Submodel	The value of this parameter specifies the desired coordinate system relative to which the position and orientation may be given. **Geodetic** – Position shall be specified as a geodetic latitude, longitude, and altitude. Orientation shall be given with respect to the reference plane shown in Figure 18, page 46. **Parent Entity** – Position and orientation shall be with respect to the entity to which the specified entity or view is attached. This value shall be invalid for top-level entities. **Submodel** – Position and orientation shall be specified with respect to the articulated part's reference coordinate system as described in Section 5.4.3. This value shall be valid only when *Object Class* is set to Articulated Part (1). Note: If *Object Class* is set to Motion Tracker (4), The coordinate system is defined by the tracking device and this parameter shall be ignored.

Parameter	Description
Object ID Type: unsigned int16 Units: N/A Values: If *Object Class* = 0: 0 – 65,535 If *Object Class* = 1: 0 – 65,535 If *Object Class* = 2: 0 – 65,535 If *Object Class* = 3: 1 – 255 If *Object Class* = 4: 0 – 255	This parameter identifies the entity, view, view group, or motion tracking device whose position is being requested. If the IG receives a Position Request packet, and if the Object ID parameter in that packet identifies a valid object, then the IG shall return a Position Response packet with the Object ID set to this value. If the IG receives a Position Request packet that identifies an invalid object, then the IG shall ignore the packet. The type of object is specified by the *Object Class* parameter. If *Object Class* is set to Articulated Part (1), this parameter shall indicate the entity whose part is identified by the *Articulated Part ID* parameter.

6.1.28 Environmental Conditions Request

At any given location, it may be impossible for the Host to determine exactly the visibility range, air temperature, or other atmospheric or surface conditions. One factor is that various IG implementations may differ in how they calculate values across transition bands and within overlapping regions. Random phenomena such as winds aloft, scud, and wave activity may also make determining instantaneous conditions at a specific point impossible.

The **Environmental Conditions Request** packet is used by the Host to request the state of the environment at a specific location. The *Request Type* parameter determines what data are returned by the IG. Each request type is represented by a power of two (i.e.,, a unique bit), so request types may be combined by adding or bit-wise ORing the values together.

The IG shall send at least one response packet for each request type specified. If the Host requests the maritime surface conditions, then the IG shall send exactly one Maritime Surface Conditions Response packet. If the Host requests weather conditions, then the IG shall send exactly one Weather Conditions Response packet. For terrestrial surface conditions requests, the IG shall respond with one **Terrestrial Surface Conditions Response** packet for each surface condition type or attribute present at the test point. If the *Request Type* parameter specifies that aerosol concentrations shall be returned, the IG shall send a **Weather Conditions Aerosol Response** packet for each weather layer that encompasses the test point.

For example, assume that two overlapping environmental regions have been defined, each containing a weather layer. The vertical ranges of the weather layers overlap, and both layers have the same layer ID. A test point is contained within both layers as illustrated in Figure 89:

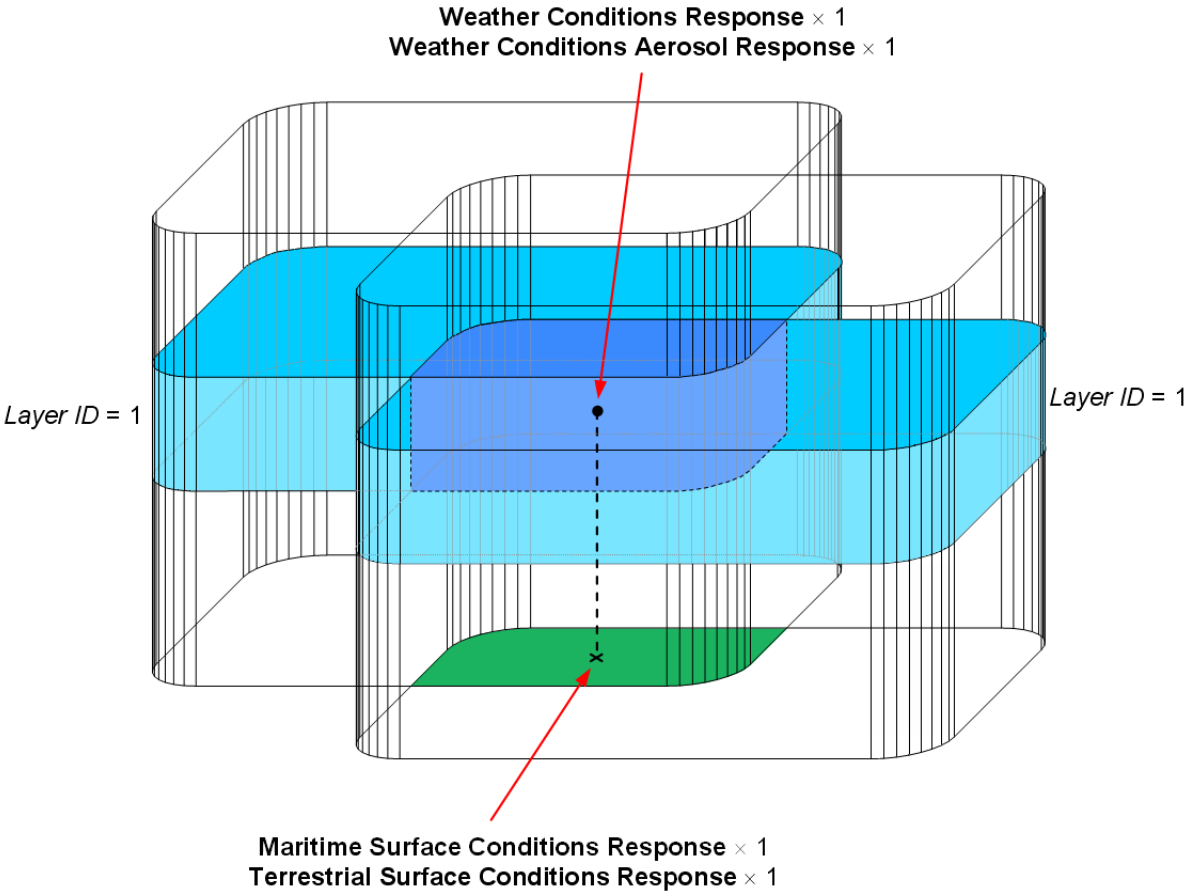


Figure 89 – Environmental Conditions Request (Weather Layers with Same ID)

The Host would send an **Environmental Conditions Request** packet to the IG, specifying the geodetic position of the test point. For this example, assume that the Host requires the weather conditions and aerosol concentrations at that point plus the terrestrial and maritime surface conditions on the terrain directly below that point. The value of the *Request Type* parameter of this packet would therefore be 15, which is the sum of the values corresponding to each request type. This is shown in Figure 90.

The IG would answer the request with each of the required response packets. Note that the IG would populate the *Request ID* parameter of each response packet with the value of the *Request ID* parameter in the **Environmental Conditions Request** packet [CIGIBP, Request IDs]. The following diagram illustrates this exchange of data between the Host and IG:

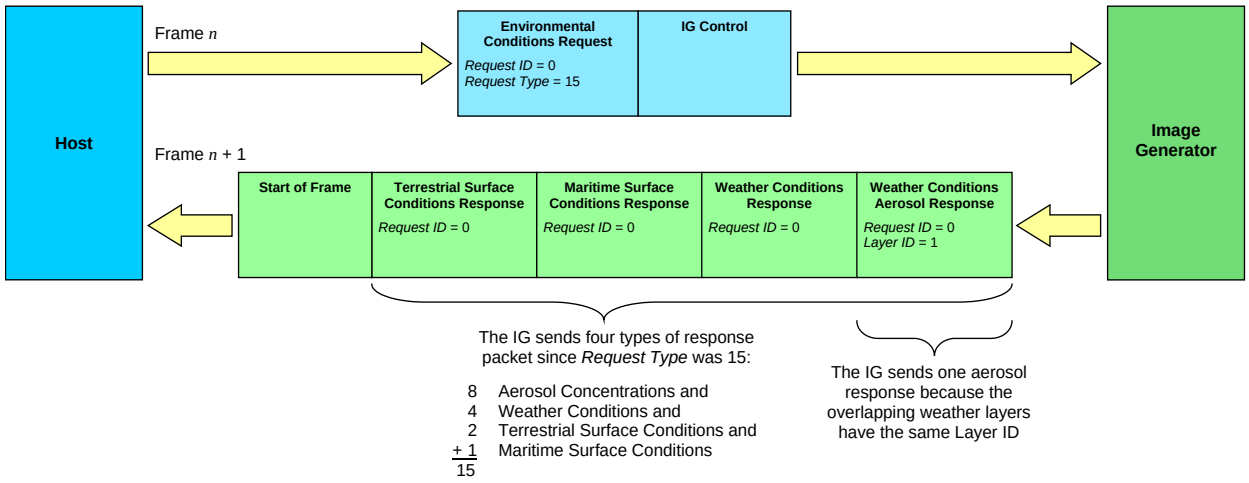


Figure 90 – Data Exchange for Environmental Conditions Request (One Aerosol)

Because the *Layer ID* of both weather layers is the same, the IG would send only one **Weather Conditions Aerosol Response** packet. This packet would contain the average (or the result of some other appropriate combining function) of the concentrations of the aerosol contained within Layer 1 of each of the two regions. In this case, the layer ID corresponds to a cloud layer, so the aerosol is liquid water. This allows for the creation of a composite weather volume in which the aerosol is more or less continuous through regions of intersection.

Alternatively, the overlapping weather layers might have different layer IDs as shown below:

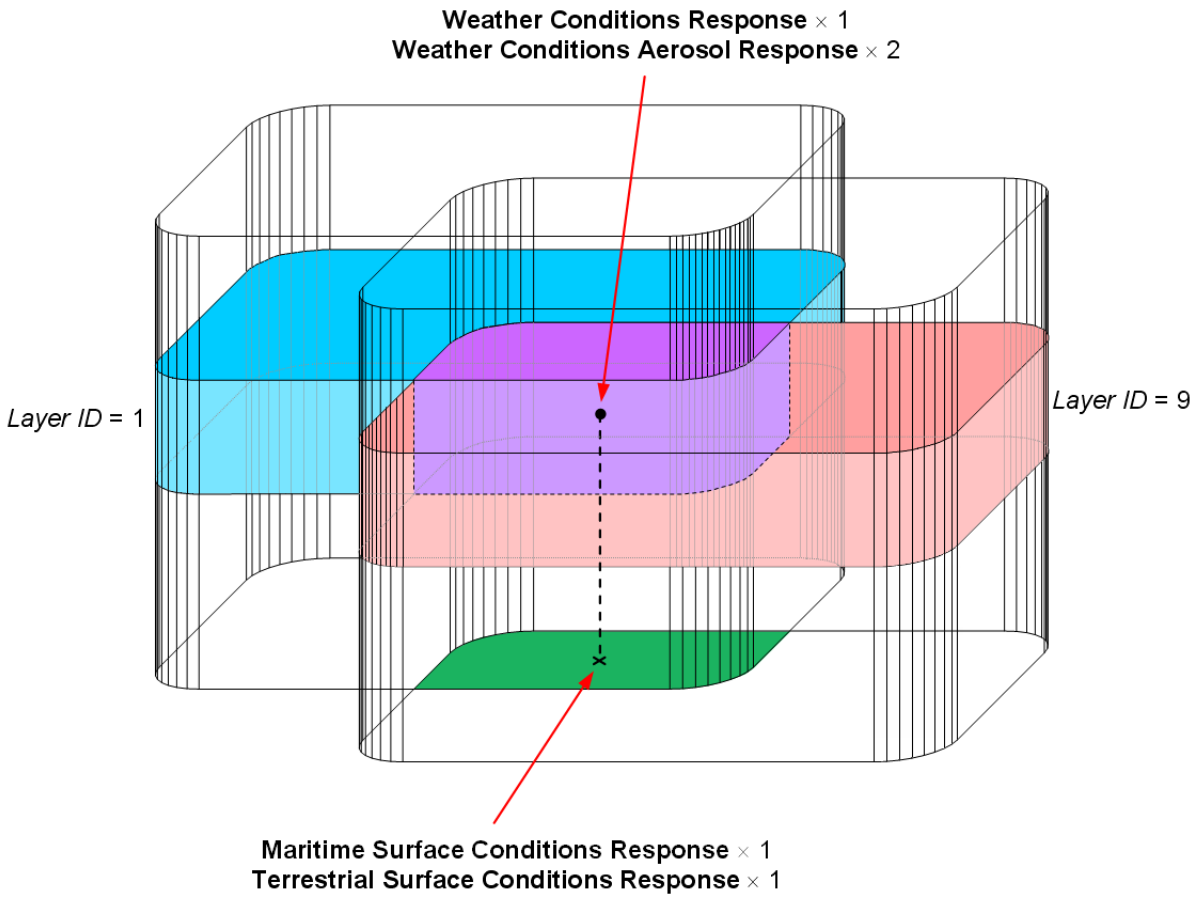


Figure 91 – Environmental Conditions Request (Weather Layers with Different IDs)

The figure above shows a cloud layer (Layer 1) overlapping with a dust layer (Layer 9). Given the same environmental conditions request, the IG would now send two **Weather Conditions Aerosol Response** packets:

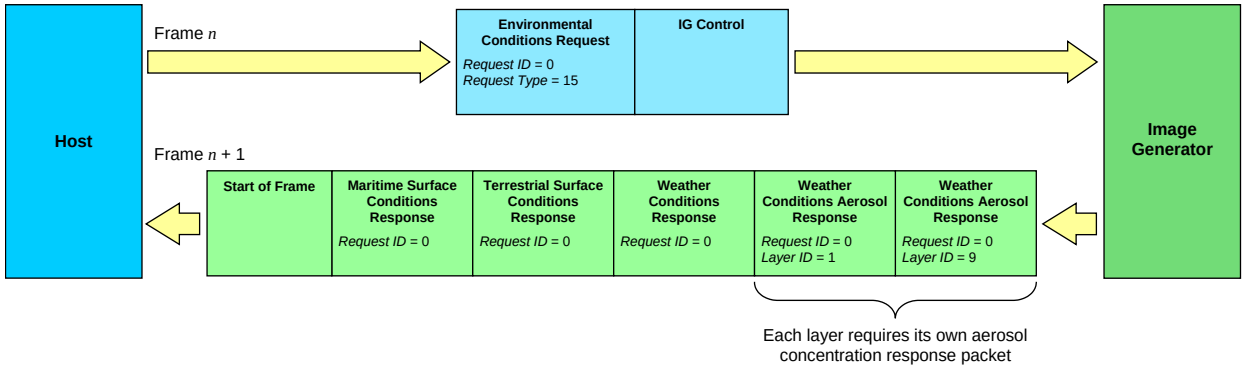


Figure 92 – Data Exchange for Environmental Conditions Request (Two Aerosols)

The *Layer ID* parameter of each response packet corresponds to the layer, thus identifying the aerosol. The concentrations of the two aerosols are independent of each other, so each layer requires its own respective **Weather Conditions Aerosol Response** packet.

The contents of the **Environmental Conditions Request** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 01Bh															
Reserved				*1				*Request ID*								Reserved															
Latitude																															
Longitude																															
Altitude																															

*1 *Request Type*

Figure 93 – Environmental Conditions Request Packet Structure

Table 34 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 34 – Environmental Conditions Request Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 32.
Packet ID Type: unsigned int16 Units: N/A Value: 01Bh	This parameter identifies this data packet as the **Environmental Conditions Request** packet. The Host shall set this parameter to 01Bh.

Parameter	Description
Request Type Type: unsigned 4-bit field Units: N/A Values: 1 Maritime Surface Conditions 2 Terrestrial Surface Conditions 4 Weather Conditions 8 Aerosol Concentrations	This parameter specifies the desired response type for the request. The numerical values listed at left may be combined by addition or bit-wise OR. The resulting value may be any combination of the following: **Maritime Surface Conditions** – The IG shall respond with a **Maritime Surface Conditions Response** packet (Section 6.2.11). **Terrestrial Surface Conditions** – The IG shall respond with a **Terrestrial Surface Conditions Response** packet (Section 6.2.12). **Weather Conditions** – The IG shall respond with a **Weather Conditions Response** packet (Section 6.2.9). **Aerosol Concentrations** – The IG shall send exactly one **Aerosol Concentration Response** packet (Section 6.2.10) for each weather layer (regardless of scope) that encompasses that location.
Request ID Type: unsigned int8 Units: N/A	This parameter identifies the environmental conditions request [CIGIBP, Request IDs]. When the IG returns a response to the request, each response packet(s) shall contain this value in its *Request ID* parameter.
Latitude Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	This parameter specifies the geodetic latitude at which the environmental state is requested. The IG shall calculate its response based on the latitude, longitude, and altitude specified in this packet.
Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	This parameter specifies the geodetic longitude at which the environmental state is requested. The IG shall calculate its response based on the latitude, longitude, and altitude specified in this packet.

Parameter	Description
Altitude Type: double float Units: meters Datum: Mean Sea Level	This parameter specifies the geodetic altitude at which the environmental state is requested. The IG shall calculate its response based on the latitude, longitude, and altitude specified in this packet. The IG shall ignore this parameter for maritime surface and terrestrial surface requests.

6.1.29 Symbol Surface Definition

The **Symbol Surface Definition** packet is used to create a symbol surface and control its position, orientation, size, and other attributes.

Each symbol surface shall be identified by a unique *Surface ID* value. When the IG receives a **Symbol Surface Definition** packet referring to a surface that does not exist, the IG shall create a new surface based on the packet's parameters. If the surface does exist, it shall be modified according to the packet's parameters.

A symbol surface shall be attached to exactly one entity or view. The *Attach Type* parameter determines the type of object to which the view shall be attached, and the *Entity ID/View ID* parameter identifies that object. If the entity or view does not exist, the **Symbol Surface Definition** packet shall be ignored.

For surfaces attached to a view, the *Left*, *Right*, *Top*, and *Bottom* parameters define the position and size of the surface. These values shall be specified relative to the view's Normalized Viewport Coordinate System as described in Section 5.4.4.3.

For surfaces attached to an entity, the *X Offset*, *Y Offset*, *Z Offset*, *Yaw*, *Pitch*, and *Roll* parameters specify the position and attitude of the surface in relation to the entity to which it is attached. The translation and rotation behavior shall be the same as for a child entity and is described in Section 5.4.4.1. The *Width* and *Height* parameters determine the size of the surface.

For billboard surfaces attached to an entity, the orientation of the entity shall have no effect on the surface's orientation; the surface shall always be parallel to the view plane as described in Section 5.4.4.2. The *Width* and *Height* parameters determine the size of the surface. The *Yaw*, *Pitch*, and *Roll* parameters shall be ignored.

Every surface has a local 2D coordinate system (UV mapping) that is used to place, rotate, and size each symbol drawn on the surface as described in Section 5.4.5.1. The *Min U*, *Max U*, *Min V*, and *Max V* parameters define this coordinate system.

The stacking order for surfaces attached to the same view is such that a surface with a lower *Surface ID* value shall be drawn behind (i.e.,, further from the eyepoint than) a surface with a higher value. For example, if three surfaces attached to the same view have *Surface ID* values of 3, 4, and 7, then Surface 3 shall be drawn first. Surface 4 shall be drawn next and may occult any overlapping areas. Finally, Surface 7 shall be drawn on top and may likewise occult parts of the other surfaces. Since surfaces attached to views are coincident with the near clipping plane, view-attached surfaces shall be drawn on top of all other objects in the view.

Any surface attached to an entity shall be subject to the normal drawing order.

Once a **Symbol Surface Definition** packet describing a symbol surface is sent to the IG, the state of that surface shall not change until another **Symbol Surface Definition** packet referencing the same Surface ID is received.

A symbol surface shall be destroyed when the Host sets the *Surface State* parameter to Destroyed (1). Any symbols associated with that surface shall also be destroyed.

If an entity is destroyed, then any symbol surfaces attached to that entity shall also be destroyed.

The contents of the **Symbol Surface Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 64																*Packet ID* = 01Ch															
Surface ID																*Entity ID/View ID*															
Reserved				*4	*3	*2	*1	Reserved																							
X Offset/Left																															
Y Offset/Right																															
Z Offset/Top																															
Yaw/Bottom																															
Pitch																															
Roll																															
Width																															
Height																															
Min U																															
Max U																															
Min V																															
Max V																															
Reserved																															

*1 *Surface State*
*2 *Attach Type*
*3 *Billboard*
*4 *Perspective Growth Enable*

Figure 94 – Symbol Surface Definition Packet Structure

Table 35 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 35 – Symbol Surface Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 64	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 64.
Packet ID Type: unsigned int16 Units: N/A Value: 01Ch	This parameter identifies this data packet as the **Symbol Surface Definition** packet. The Host shall set this parameter to 01Ch.
Surface ID Type: unsigned int16 Units: N/A Values: N/A	This parameter specifies the symbol surface to which this packet shall be applied.

Parameter	Description
Entity ID/View ID Type: unsigned int16 Units: N/A	This parameter specifies the entity or view to which this surface shall be attached.
Surface State Type: 1-bit field Units: N/A Values: 0 Active 1 Destroyed	This parameter specifies whether the symbol surface may be active or destroyed. **Active** – The surface shall be active and symbols may be drawn on it. The surface may be positioned, oriented, and sized; and it may be attached to an entity or a view. **Destroyed** – The surface shall be removed from the system. Any symbols drawn to it shall also be destroyed. All other state parameters in this packet shall be ignored.
Attach Type Type: 1-bit field Units: N/A Values: 0 Entity 1 View	This parameter specifies whether the surface may be attached to an entity or a view. **Entity** – The surface shall be attached to the entity specified by the *Entity ID/View ID* parameter. **View** – The surface shall be attached to the view specified by the *Entity ID/View ID* parameter. If the specified entity or view does not exist, this packet shall be ignored.
Billboard Type: 1-bit field Units: N/A Values: 0 Non-Billboard 1 Billboard	This parameter specifies whether an entity-attached surface is treated as a billboard. **Non-Billboard** – The surface shall be oriented in relation to the entity's local coordinate system by the *Yaw*, *Pitch*, and *Roll* parameters **Billboard** – The surface shall be oriented such that the normal vector from the center of the surface is parallel to the viewing vector as shown in Figure 25. If the surface is attached to a view, then the IG shall ignore this parameter.

Parameter	Description
Perspective Growth Enable Type: 1-bit field Units: N/A Values: 0 Disabled 1 Enabled	This parameter specifies whether the surface may appear to maintain a constant size or have perspective growth as the entity to which the surface is attached moves closer to or further from the eyepoint. If the surface is attached to an entity and is a billboard, and if this parameter is set to Disabled (0), then the surface shall appear to stay the same size (i.e.,, shall cover the same area of the view) regardless of its distance from the eyepoint. If the surface is attached to an entity and is a billboard, and if this parameter is set to Enabled (1), then the surface shall appear to change size relative to the viewport as the entity to which the surface is attached moves away from or closer to the eyepoint. If the surface is attached to an entity but is not a billboard, then the IG shall ignore this parameter. If the surface is attached to a view, then the IG shall ignore this parameter.
X Offset (Entity-attached surfaces) Type: single float Units: meters Datum: Entity's local coordinate system	For a non-billboard surface attached to an entity, the IG shall place the surface at this distance along the entity's **X** axis from the entity's reference point to the center of the surface (see Section 5.4.4.1). For a billboard surface attached to an entity, the IG shall place the surface at this distance along the surface's **X** axis from the center of the surface to the entity's reference point (see Section 5.4.4.2).
Left (View-attached surfaces) Type: single float Units: viewport width Datum: Normalized viewport coordinate system	For a surface attached to a view, the IG shall place the surface at this distance from the left edge of the viewport to the surface's leftmost boundary. This distance shall be measured as a fraction of the viewport's width (see Section 5.4.4.3).

Parameter	Description
Y Offset (Entity-attached surfaces) Type: single float Units: meters Datum: Entity's local coordinate system	For a non-billboard surface attached to an entity, the IG shall place the surface at this distance along the entity's **Y** axis from the entity's reference point to the center of the surface (see Section 5.4.4.1). For a billboard surface attached to an entity, the IG shall place the surface at this distance along the surface's **Y** axis from the center of the surface to the entity's reference point (see Section 5.4.4.2).
Right (View-attached surfaces) Type: single float Units: viewport width Datum: Normalized viewport coordinate system	For a surface attached to a view, the IG shall place the surface at this distance from the left edge of the viewport to the surface's rightmost boundary. This distance shall be measured as a fraction of the viewport's width (see Section 5.4.4.3).
Z Offset (Entity-attached surfaces) Type: single float Units: meters Datum: Entity's local coordinate system	For a non-billboard surface attached to an entity, the IG shall place the surface at this distance along the entity's **Z** axis from the entity's reference point to the center of the surface (see Section 5.4.4.1). For a billboard surface attached to an entity, the IG shall place the surface at this distance along the surface's **Z** axis from the center of the surface to the entity's reference point (see Section 5.4.4.2).
Top (View-attached surfaces) Type: single float Units: viewport height Datum: Normalized viewport coordinate system	For a surface attached to a view, the IG shall place the surface at this distance from the bottom edge of the viewport to the surface's topmost boundary. This distance shall be measured as a fraction of the viewport's height (see Section 5.4.4.3).

Parameter	Description
Yaw (Entity-attached surfaces) Type: single float Units: degrees Values: 0.0 – 360.0 Datum: Surface's reference plane	For a non-billboard surface attached to an entity, the IG shall orient the surface about its **Z** axis as described in Section 5.4.4.1 For entity-attached billboard surfaces, this parameter shall be ignored.
Bottom (View-attached surfaces) Type: single float Units: viewport height Datum: Normalized viewport coordinate system	For a surface attached to a view, the IG shall place the surface at this distance from the bottom edge of the viewport to the surface's bottommost boundary. This distance shall be measured as a fraction of the viewport's height (see Section 5.4.4.3).
Pitch Type: single float Units: degrees Values: -90.0 – 90.0 Datum: Surface's reference plane	For a non-billboard surface attached to an entity, the IG shall orient the surface about its **Y** axis as described in Section 5.4.4.1 For entity-attached billboard surfaces, the IG shall ignore this parameter. For a surface attached to a view, this parameter shall be ignored.
Roll Type: single float Units: degrees Values: -180.0 – 180.0 Datum: Surface's reference plane	For a non-billboard surface attached to an entity, the IG shall orient the surface about its **X** axis as described in Section 5.4.4.1 For entity-attached billboard surfaces, the IG shall ignore this parameter. For a surface attached to a view, the IG shall ignore this parameter.

Parameter	Description
Width Type: single float Units: Non-billboard, Entity-attached: meters Billboard, Entity-attached, Perspective growth enabled: meters Billboard, Entity-attached, Perspective growth disabled: degrees Values: For meters: > 0.0 For degrees: > 0.0 to < 180.0	If the surface is attached to an entity and is not a billboard, then the IG shall set the width of the surface to the specified number of meters. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Enabled (1), then the IG shall set the width of the surface to the specified number of meters. The apparent size of the surface shall depend upon the distance to the surface from the eyepoint. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Disabled (0), then the IG shall set the width of the surface to the specified angle. The occupied view space shall remain constant regardless of the surface's distance from the eyepoint. If the surface is attached to a view, then the IG shall ignore this parameter.
Height Type: single float Units: Non-billboard, Entity-attached: meters Billboard, Entity-attached, Perspective growth enabled: meters Billboard, Entity-attached, Perspective growth disabled: degrees Values: For meters: > 0.0 For degrees: > 0.0 to < 180.0	If the surface is attached to an entity and is not a billboard, then the IG shall set the height of the surface to the specified number of meters. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Enabled (1), then in the IG shall set the height of the surface to the specified number of meters. The apparent size of the surface shall depend upon the distance to the surface from the eyepoint. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Disabled (0), then the IG shall set the height of the surface to the specified angle. The occupied view space shall remain constant regardless of the surface's distance from the eyepoint. If the surface is attached to a view, then the IG shall ignore this parameter.
Min U Type: single float Units: Symbol surface horizontal units Values: < *Max U*	This parameter specifies the U coordinate that shall correspond to the leftmost boundary of the symbol surface. Symbol surface 2D coordinate systems and horizontal units are described in Section 5.4.5.1.

Parameter	Description
Max U Type: single float Units: Symbol surface horizontal units Values: > *Min U*	This parameter specifies the U coordinate that shall correspond to the rightmost boundary of the symbol surface. Symbol surface 2D coordinate systems and horizontal units are described in Section 5.4.5.1.
Min V Type: single float Units: Symbol surface vertical units Values: < *Max V*	This parameter specifies the V coordinate that shall correspond to the bottommost boundary of the symbol surface. Symbol surface 2D coordinate systems and vertical units are described in Section 5.4.5.1.
Max V Type: single float Units: Symbol surface vertical units Values: > *Min V*	This parameter specifies the V coordinate that shall correspond to the topmost boundary of the symbol surface. Symbol surface 2D coordinate systems and vertical units are described in Section 5.4.5.1.

6.1.30 Symbol Text Definition

The **Symbol Text Definition** packet is used to define a string of text, as well as its alignment, orientation, font, and size.

Each text symbol is identified by a *Symbol ID* value that is unique from all other symbols (text or otherwise). Every symbol shall be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

Once a **Symbol Text Definition** packet describing a text symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Circle Definition**, **Symbol Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing text symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

The *Font ID* parameter uniquely identifies a specific font and is defined by the IG. A font is a unique combination of typeface, style (such as italic), and weight (such as bold). Therefore, any special attribute of the font such as bold or italics is identified using a separate *Font ID*. Table 36 defines several default font styles; however, the exact typeface is IG-dependent. All other font assignments are IG-defined.

Font size is defined as the vertical space that a font occupies. This includes the cap height as well as the heights of any ascenders, descenders, accent marks, and vertical padding.

The text string shall be composed of UTF-8 character data. The text shall be terminated by NULL, or zero (0). If the terminating byte is not the last byte before an eight-byte boundary, then the remainder of the packet shall be padded with zeroes up to the next eight-byte boundary. Zero-length text strings shall be terminated with four bytes containing NULL to maintain eight-byte alignment. The maximum text length is dependent upon the sizes of the individual UTF-8 character data.

The *Packet Size* parameter shall contain the number of bytes up to and including the *Font Size* parameter (a total of 12 bytes) and the total number of bytes within the text, including the terminating NULL and any padding. This value shall be an even multiple of eight (8). For example, if the string "Hello World!" were sent to the IG, the packet size would be 12 + 24, or 32 bytes. Figure 95 illustrates the byte allocation for the packet:

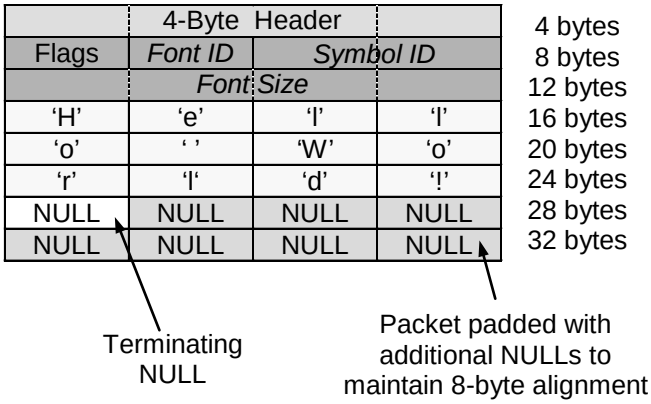
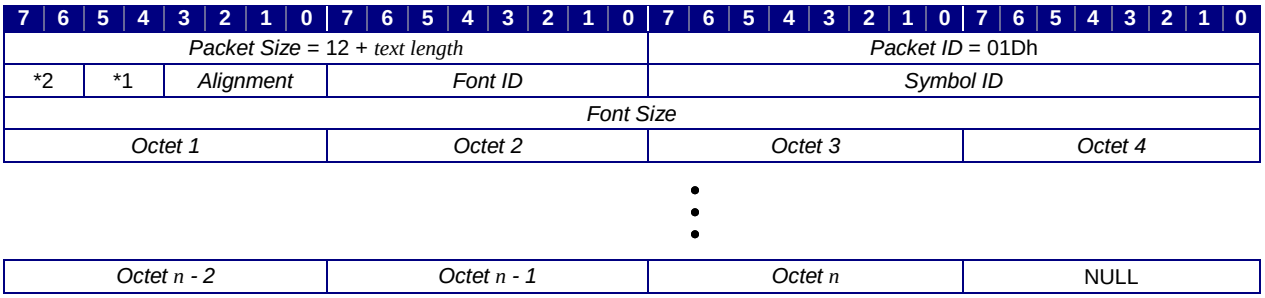


Figure 95 – Example of Symbol Text Definition Packet

When the IG creates a new symbol, that symbol shall be hidden by default. The symbol shall not be made visible until the Host sends a **Symbol Control** packet and sets the *Symbol State* parameter set to Visible (1).

The contents of the **Symbol Text Definition** packet are as follows:



*1 *Orientation*
*2 Reserved

Figure 96 – Symbol Text Definition Packet Structure

Table 36 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 36 – Symbol Text Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 16 ≤ (12 + *text length*) ≤ 65528 where *text length* is the number of characters, including NULL characters, described by this packet	This parameter indicates the number of bytes in this data packet. The Host shall set this size using the formula in the left column. This value shall be divisible by eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 01Dh	This parameter identifies this data packet as the **Symbol Text Definition** packet. The Host shall set this parameter to 01Dh.

Parameter	Description
Alignment Type: unsigned 4-bit field Units: N/A Values: 0 Top Left 1 Top Center 2 Top Right 3 Center Left 4 Center 5 Center Right 6 Bottom Left 7 Bottom Center 8 Bottom Right	This parameter specifies the position at which the IG shall place the symbol's reference point in relation to the text. This parameter also determines whether the text shall be left-, center-, or right-justified. The following example shows each of the alignment points with a red dot: 
Orientation Type: unsigned 2-bit field Units: N/A Values: 0 Left to Right 1 Top to Bottom 2 Right to Left 3 Bottom to Top	This parameter specifies the orientation of the text. The IG shall render characters in the direction specified by this parameter. For a text string "ABC", specifying an orientation of Left to Right (0) would produce the following: ABC An orientation of Top to Bottom (1) would look like this: A B C An orientation of Right to Left (2) would look like this: CBA An orientation of Bottom to Top (3) would look like this: C B A

Parameter	Description
Font ID Type: unsigned int8 Units: N/A Values: 0 IG default <u>Proportional:</u> 1 Sans Serif 2 **Sans Serif Bold** 3 Sans Serif *Italic* 4 ***Sans Serif Bold Italic*** 5 Serif 6 **Serif Bold** 7 Serif *Italic* 8 ***Serif Bold Italic*** <u>Monospace:</u> 9 Sans Serif 10 **Sans Serif Bold** 11 *Sans Serif Italic* 12 ***Sans Serif Bold Italic*** 13 Serif 14 **Serif Bold** 15 *Serif Italic* 16 ***Serif Bold Italic*** 17–255 IG-defined	This parameter specifies the font that shall be used for this text symbol. This document defines a set of default proportional (variable-width) and monospace (constant-width) font styles for compatibility. The IG shall provide styles for this default set; however, the exact typefaces used shall be IG-dependent. Font IDs 17 through 255 are entirely IG-defined.
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a symbol with the specified identifier already exists, and if that symbol is of a type other than text, then the IG shall destroy the existing symbol and any children and shall create a new symbol.
Font Size Type: single float Units: Symbol surface vertical units	This parameter specifies the font size, This includes the cap height as well as the heights of any ascenders, descenders, accent marks, and vertical padding. This size shall be defined in terms of the vertical units defined by the symbol surface's 2D coordinate system (see Section 5.4.5.1).
Octet n Type: octet Units: N/A	These 8-bit data elements are used to store the UTF-8 code points in the string. The maximum length of the string, including a terminating NULL, is 65516 bytes.

6.1.31 Symbol Circle Definition

The **Symbol Circle Definition** packet is used to create a single circle or arc. This packet may also be used to create a composite symbol composed of circles and/or arcs. Note that this section uses the term "circle" to refer to both circles and arcs unless otherwise indicated.

Each circle symbol is identified by a *Symbol ID* value that is unique from all other symbols (including text and line symbols). Every symbol shall be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

Once a **Symbol Circle Definition** packet describing a circle symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Textured Circle Definition**, **Symbol Polygon Definition**, **Symbol Textured Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

The center of each circle shall be located at a point (u, v) on the symbol's 2D coordinate system (see Section 5.4.5.2) as defined by the *Center U* and *Center V* parameters. The radius of the circle shall be specified in scaled symbol surface units by the *Radius* parameter. Note that if the symbol surface's 2D coordinate system is defined such that horizontal units are not the same length as vertical units, then a circle shall appear as an ellipse and an arc shall appear as an elliptical arc.

The *Start Angle* and *End Angle* parameters define the endpoints of the curve. These angles shall be measured counter-clockwise from the symbol's +**U** axis as shown below in Figure 97 and Figure 98. If these two values are equal, then the symbol shall define a full circle. If these two values are not equal, then the symbol shall define an arc. For circles, it is recommended that values of 0.0 be used for consistency and to avoid floating-point errors.

A circle may either be outlined or filled, depending upon the value of the *Drawing Style* parameter.

A curved line's pen attributes are defined by the *Line Width*, *Stipple Pattern*, and *Stipple Pattern Length* parameters.

Line Width specifies the thickness of the line in scaled symbol surface units. Note that if the surface's horizontal and vertical units are not equal in size, then curved lines shall not appear to be uniform in thickness.

The *Stipple Pattern* parameter defines a bit mask to be applied to the line: if a bit is set (1) then the section of the curve corresponding to that bit shall be drawn; if the bit is cleared (0) then the corresponding section shall not be drawn.

The length of each section shall be equal to $^1/_{16}$ of the length specified by the *Stipple Pattern Length* parameter. This parameter defines the length of the stipple pattern in terms of scaled symbol surface units. If the curved line is longer than the stipple pattern length, then the pattern shall be repeated.

Figure 97 shows an example of an arc drawn as a curved line. The arc is centered at (0, 0) and has a start angle and end angle of 45° and 135°, respectively. Because the curve is longer than the stipple pattern length, the stipple pattern is repeated.

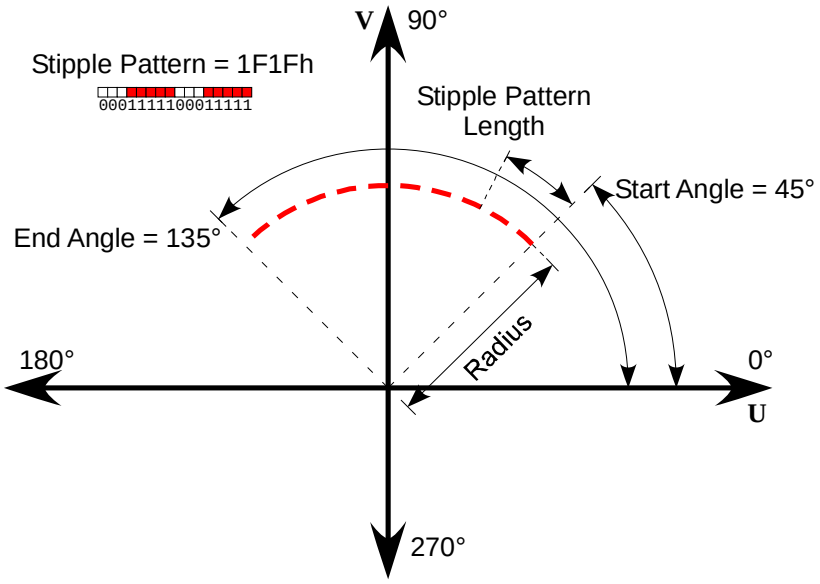


Figure 97 – Example of Line Circle (Arc) Symbol

Note that the end-cap style of a curved line is implementation-dependent and may optionally be controlled with a **Component Control** packet.

If *Drawing Style* is set to Fill (1), then the circle or sector is drawn as a filled region defined by the *Start Angle*, *End Angle*, *Radius*, and *Inner Radius* parameters. Note that if the *Inner Radius* parameter is 0.0, then the entire circle or sector shall be filled. If *Inner Radius* is equal or greater than the outer radius, then no circle shall be drawn.

The *Line Width*, *Stipple Pattern*, and *Stipple Pattern Length* parameters shall be ignored for filled circle symbols.

Figure 98 shows an example of a partially filled sector centered at (0, 0) with a start angle and end angle of 45° and 135°, respectively. The radial width of the filled region is determined by the *Radius* and *Inner Radius* parameters.

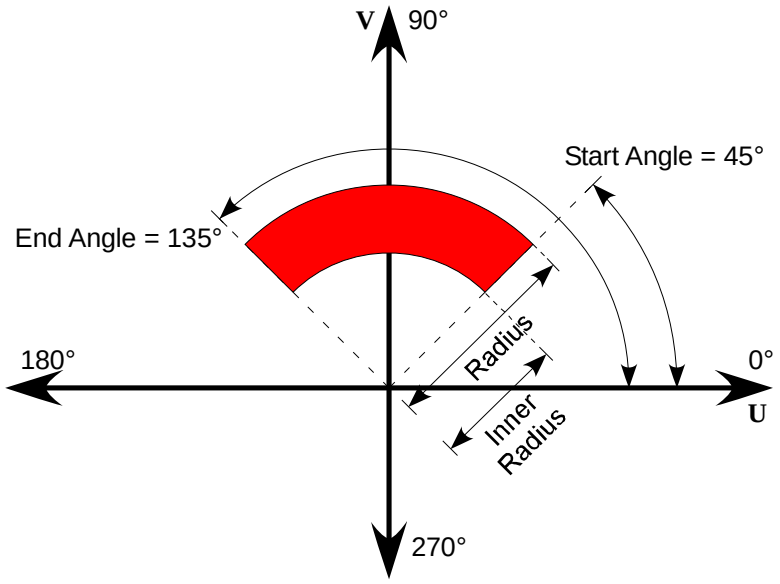


Figure 98 – Example of a Filled Circle (Arc) Symbol

Note that all circles and arcs shall be drawn counter-clockwise. If an arc's *Start Angle* is greater than its *End Angle*, then the arc shall cross the +**U** axis as shown in Figure 99:

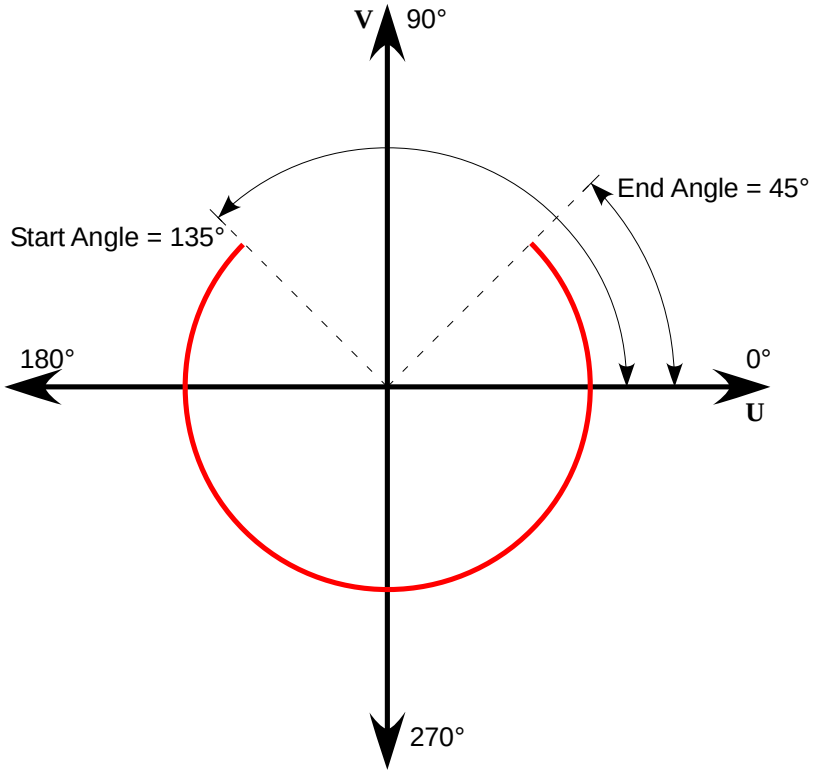


Figure 99 – Example of an Arc Crossing the +U Axis

When the IG creates a new symbol, that symbol shall always be hidden by default. The symbol shall not be made visible until the Host sends a **Symbol Control** packet and sets the *Symbol State* parameter set to Visible (1).

The contents of the **Symbol Circle Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24 + (24 × *n*)																*Packet ID* = 01Eh															
Symbol ID																*Stipple Pattern*															
Reserved								*1								Reserved															
Line Width																															
Stipple Pattern Length																															
Reserved																															
Center U_1																															
Center V_1																															
Radius$_1$																															
Inner Radius$_1$																															
Start Angle$_1$																															
End Angle$_1$																															
⋮																															
Center U_n																															
Center V_n																															
Radius$_n$																															
Inner Radius$_n$																															
Start Angle$_n$																															
End Angle$_n$																															

*1 *Drawing Style*

Figure 100 – Symbol Circle Definition Packet Structure

Table 37 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 37 – Symbol Circle Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 24 ≤ [24 + (24 × *n*)] ≤ 65520 where *n* is the number of circles described by this packet	This parameter specifies the number of bytes in this data packet. The Host shall set this size using the formula in the left column. This value shall be divisible by eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 01Eh	This parameter identifies this data packet as the **Symbol Circle Definition** packet. The Host shall set this parameter to 01Eh.

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a circle symbol with the specified identifier already exists, then the IG shall replace the existing symbol geometry and shall retain the existing state. If a symbol with the specified identifier already exists, and if that symbol is of a type other than a circle or arc, then the IG shall destroy the existing symbol and any children and shall create a new symbol. Note that the Host shall subsequently send a **Symbol Control** packet to set the new symbol's visibility and state.
Stipple Pattern Type: unsigned int16 Units: N/A	This parameter specifies the dash pattern to be used when drawing the curved line of a circle or arc. Each curved line is divided into sections that are $^{1}/_{16}$ of the length specified by the *Stipple Pattern Length* parameter. If a bit is set (1) then the IG shall draw the line section corresponding to that bit; if the bit is cleared (0) then the IG shall not draw the corresponding section. If the value of this parameter is 0xFFFF, then the IG shall draw a solid line. If the value is 0x00, then the IG shall not draw the line. If the line is longer than the stipple pattern length, then the IG shall repeat the pattern The IG shall ignore this parameter if *Drawing Style* is set to Fill (1).
Drawing Style Type: 1-bit field Units: N/A Values: 0 Line 1 Fill	This parameter specifies whether the circles and arcs defined in this packet are defined as curved lines or filled volumes. If the Host sets this parameter to Line (0), then the IG shall draw the arc or circumference of the circle. If the Host sets this parameter to Fill (1), then the IG shall fill the region between the inner and outer radii.

Parameter	Description
Line Width Type: single float Units: Scaled symbol surface units	This parameter specifies the line thickness that the IG shall use when drawing the circles and arcs defined in this packet. This thickness shall be measured in symbol surface units and shall be scaled if the symbol is scaled (see Section 5.4.5.2). If the symbol surface's horizontal and vertical units are not the same size, then horizontal, diagonal, and vertical lines shall not appear to be the same thickness. The IG shall ignore this parameter if *Drawing Style* is set to Fill (1).
Stipple Pattern Length Type: single float Units: Scaled symbol surface units	This parameter specifies the length of one complete repetition of the stipple pattern. This length shall be measured in symbol surface units and shall be scaled if the symbol is scaled (see Section 5.4.5.2). If a line is longer than the stipple pattern length, then the IG shall repeat the pattern along that line. The IG shall ignore this packet if the *Drawing Style* parameter is set to Fill (1).
Center U Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the u position of the circle's center. The IG shall place the center with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Center V Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the v position of the circle's center. The IG shall place the center with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Radius Type: single float Units: Scaled symbol surface units Values: ≥ 0.0 Datum: Center of circle	This parameter specifies the radius of the outer circumference of the circle or arc. This value shall be measured in scaled symbol surface units (see Section 5.4.5.2). If *Drawing Style* is set to Fill (1), then the IG shall create a filled region bounded by the inner and outer radii and the start and end angles as illustrated in Figure 98. If *Drawing Style* is set to Line (0), then the IG shall draw the circumference with the line width centered on the radius.

Parameter	Description
Inner Radius Type: single float Units: scaled symbol surface units Value: ≥ 0.0 to < *Radius* Datum: Center of circle	This parameter specifies the inner radius of a filled circle or arc. This value shall be measured in scaled symbol surface units (see Section 5.4.5.2). If *Drawing Style* is set to Fill (1), then the IG shall create a filled region bounded by the inner and outer radii and the start and end angles. The IG shall ignore this parameter if *Drawing Style* is set to Line (0).
Start Angle Type: single float Units: degrees Value: 0.0 – 360.0 Datum: Symbol's +**U** axis	This parameter specifies the starting angle of the arc. This angle shall be measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc shall cross the +**U** axis.
End Angle Type: single float Units: scaled symbol surface units Value: 0.0 – 360.0 Datum: Symbol's +**U** axis	This parameter specifies the ending angle of the arc. This angle shall be measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc shall cross the +**U** axis.

6.1.32 Symbol Polygon Definition

The **Symbol Polygon Definition** packet is used to define a set of line segments or points. This packet may be used to create points, lines, a line strip, a line loop, triangles, a triangle strip, or a triangle fan. Note that this section includes all of these primitives when referring to "line symbols."

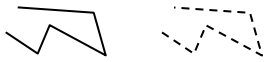
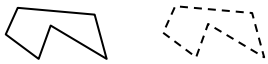
Each line symbol is identified by a *Symbol ID* value that is unique from all other symbols (including text and circle symbols). Every symbol shall be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

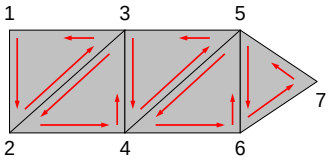
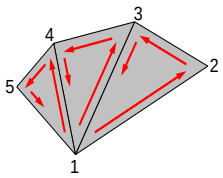
Once a **Symbol Polygon Definition** packet describing a circle symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Circle Definition**, **Symbol Textured Circle Definition**, **Symbol Textured Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

Every line symbol is defined as an ordered set of zero or more points. Each point shall be defined with respect to the symbol's 2D coordinate system (see Section 5.4.5.2) by a pair of coordinates specified in the *Vertex U* and *Vertex V* parameters.

The method and order by which the points are connected is determined by the *Primitive Type* parameter. These primitives are described in Table 38:

Table 38 – Line Symbol Primitive Types

Primitive Type	Description	Example
Point	Zero or more points, each at some location (u, v) as defined by a *Vertex U* and *Vertex V* parameter pair.	
Line	Zero or more lines drawn between each pair of odd- and even-numbered vertex. Each vertex is used exactly one time. If the packet contains an odd number of vertices, then the last vertex shall be disregarded.	
Line Strip	A contiguous series of line segments drawn between each pair of consecutive vertices. The first and last vertices are used exactly one time; all others are used exactly twice.	
Line Loop	A closed Line Strip that includes a line segment connecting the first and last vertices. Each vertex is used exactly twice.	

Primitive Type	Description	Example
Triangle	Zero or more triangles formed from three consecutive vertices. Each vertex is used to define exactly one triangle. If the number of vertices defined in the packet is not divisible by three, then the remaining vertices shall be disregarded. The IG shall draw front-facing triangles, which are defined as those whose vertices are arranged counter-clockwise from the perspective of the view's eyepoint. The IG shall not draw triangles whose vertices are wound clockwise or are coincident.	
Triangle Strip	A connected series of filled triangles formed from a series of vertices. The first three vertices form the first triangle in the triangle strip. Each successive vertex forms a triangle with the previous two vertices. For each triplet, the first vertex of the triangle is always the lowest odd-numbered vertex, followed by the lowest even-numbered vertex, and then the highest-numbered vertex. The IG shall draw front-facing triangles, which are defined as those whose vertices are arranged counter-clockwise from the perspective of the view's eyepoint. The IG shall not draw triangles whose vertices are wound clockwise or are coincident.	 Arrows show order (and direction) in which vertices are connected. Note: Black outlines are for illustrative purposes only and are not drawn as part of the triangle strip.
Triangle Fan	A connected series of filled triangles formed from a series of vertices. Every triangle in the fan uses the first vertex. The first three vertices form the first triangle. For each successive vertex v_n, a triangle is formed between vertex v_1, vertex v_{n-1}, and v_n. The IG shall draw front-facing triangles, which are defined as those whose vertices are arranged counter-clockwise from the perspective of the view's eyepoint. The IG shall not draw triangles whose vertices are wound clockwise or are coincident.	 Arrows show order (and direction) in which vertices are connected. Note: Black outlines are for illustrative purposes only and are not drawn as part of the triangle fan.

The pen attributes of each line comprising a line symbol are defined by the *Line Width*, *Stipple Pattern*, and *Stipple Pattern Length* parameters.

Line Width specifies the thickness of the line in scaled symbol surface units. Note that if the surface's horizontal and vertical units are not equal in size, then horizontal, diagonal, and vertical lines shall not appear to be the same thickness.

The *Stipple Pattern* parameter defines a bit mask to be applied to the line: if a bit is set (1) then the section of the line corresponding to that bit shall be drawn; if the bit is cleared (0) then the corresponding section shall not be drawn. If the value of this parameter is 0xFFFF, then the line is solid.

The length of each section shall be equal to $^{1}/_{16}$ of the length specified by the *Stipple Pattern Length* parameter. This parameter defines the length of the stipple pattern in terms of scaled symbol surface units. If the line is longer than the stipple pattern length, then the pattern shall be repeated.

The pen attributes shall be ignored for triangles, triangle strips, and triangle fans.

The contents of the **Symbol Polygon Definition** packet are as follows:

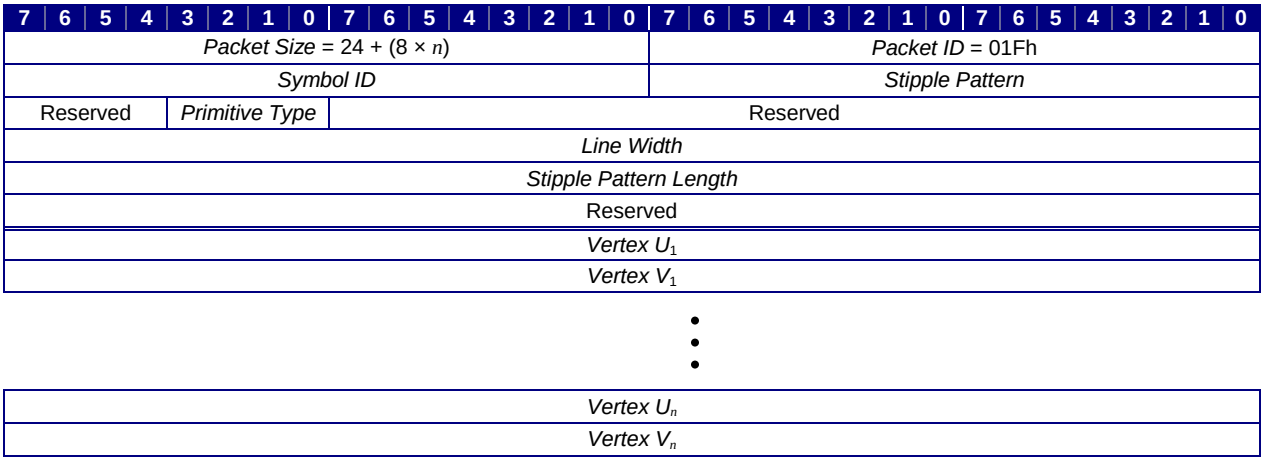


Figure 101 – Symbol Polygon Definition Packet Structure

Table 39 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 39 – Symbol Polygon Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: $24 \le [24 + (8 \times n)] \le 65528$ where n is the number of vertices described by this packet	This parameter indicates the number of bytes in this data packet. The Host shall set this size using the formula in the left column. This value shall be divisible by eight (8). The minimum size of the **Symbol Polygon Definition** packet is 16 bytes.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 01Fh	This parameter identifies this data packet as the **Symbol Polygon Definition** packet. The Host shall set this parameter to 01Fh .
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a symbol with the specified identifier already exists, and if that symbol is of a type other than a line primitive, then the IG shall destroy the existing symbol and any children and shall create a new symbol.
Stipple Pattern Type: unsigned int16 Units: N/A	This parameter specifies the dash pattern used when drawing lines. Each line is divided into sections that are $^{1}/_{16}$ of the length specified by the *Stipple Pattern Length* parameter. If a bit is set (1) then the IG shall draw the line section corresponding to that bit; if the bit is cleared (0) then the IG shall not draw the corresponding section. If the value of this parameter is 0xFFFF, then the IG shall draw a solid line. If the value is 0x00, then the IG shall not draw the line. If the line is longer than the stipple pattern length, then the IG shall repeat the pattern. The IG shall ignore this parameter if the *Primitive Type* parameter is set to Point (0), Triangle (4), Triangle Strip (5), or Triangle Fan (6).

Parameter	Description
Primitive Type Type: 4-bit field Units: N/A Values: 0 Point 1 Line 2 Line Strip 3 Line Loop 4 Triangle 5 Triangle Strip 6 Triangle Fan	This parameter specifies the type of point or line primitive defined by this packet. Each primitive type is described in Table 38.
Line Width Type: single float Units: Scaled symbol surface units	This parameter specifies the line thickness that the IG shall use when drawing the primitives defined in this packet. This thickness shall be measured in symbol surface units and shall be scaled if the symbol is scaled (see Section 5.4.5.2). For point primitives, this parameter shall specify the diameter of each point in the symbol. For line, line strip, and line loop primitives, this parameter shall specify the thickness of each line in the symbol. If the symbol surface's horizontal and vertical units are not the same size, then horizontal, vertical, and diagonal lines shall not appear to be the same thickness. The IG shall ignore this parameter if the *Primitive Type* parameter is set to Triangle (4), Triangle Strip (5), or Triangle Fan (6).
Stipple Pattern Length Type: single float Units: Scaled symbol surface units	This parameter specifies the length of one complete repetition of the stipple pattern. This length shall be measured in symbol surface units and shall be scaled if the symbol is scaled (see Section 5.4.5.2). If a line is longer than the stipple pattern length, then the IG shall repeat the pattern along that line. The IG shall ignore this parameter if the *Primitive Type* parameter is set to Triangle (4), Triangle Strip (5), or Triangle Fan (6).

Parameter	Description
Vertex U Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the u position of a vertex. The IG shall place this vertex with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Vertex V Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the v position a vertex. The IG shall place this vertex with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).

6.1.33 Symbol Clone

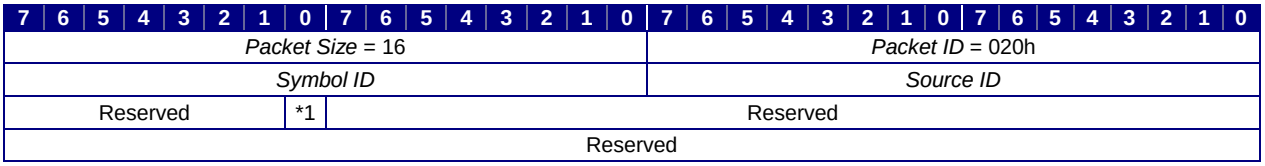
The **Symbol Clone** packet may be used to create an exact copy of a symbol. The copy shall inherit all attributes that were defined by the **Symbol Text Definition**, **Symbol Circle Definition**, **Symbol Polygon Definition**, or **Symbol Clone** packet that was used to create the original symbol. Any operations that are performed upon the copy (i.e., translation, rotation, or change of color) shall not affect the original unless otherwise dictated by a hierarchical relationship. Conversely any operations that are performed upon the original (i.e., translation, rotation, or change of color) shall not affect the copy unless otherwise dictated by a hierarchical relationship.

Alternatively, the **Symbol Clone** packet may be used to instantiate an IG-defined symbol template (see Section 5.3.3). Operations performed on the symbol instance shall not affect the template.

Once a **Symbol Clone** packet is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Circle Definition**, **Symbol Textured Circle Definition**, **Symbol Polygon Definition**, **Symbol Textured Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

When a new symbol is created with a **Symbol Clone** packet, that symbol is hidden by default. The symbol shall not be made visible until the Host sends a **Symbol Control** packet and sets the *Symbol State* parameter set to Visible (1).

The contents of the **Symbol Clone** packet are as follows:



*1 *Source Type*

Figure 102 – Symbol Clone Packet Structure

Table 40 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 40 – Symbol Clone Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 16	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 16.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 020h	This parameter identifies this data packet as the **Symbol Clone** packet. The Host shall set this parameter to 020h.
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a symbol with the specified identifier already exists, and if that symbol is of a different type, then the IG shall destroy the existing symbol and any children and shall create a new symbol.
Source ID Type: unsigned int16 Units: N/A	This parameter identifies the symbol to be copied or the symbol template to be instantiated. If *Source Type* is set to Symbol (0), then the IG shall create a new symbol that is identical to the one specified by this parameter. If *Source Type* is set to Symbol Template (1), then the IG shall create a new symbol from the template specified by this parameter. If the specified source does not exist, then the IG shall ignore this packet.
Source Type Type: 1-bit field Units: N/A Values: 0 Symbol 1 Symbol Template	This parameter determines whether the new symbol may be a copy of an existing symbol or an instance of an IG-defined symbol template. If this parameter is set to Symbol (0), then the IG shall create a new symbol that is identical to the one identified by *Source ID*. If this parameter is set to Symbol Template (1), then the IG shall create a new symbol from the template identified by *Source ID*.

6.1.34 Symbol Control

The **Symbol Control** packet is used to specify position, rotation, and other attributes describing a symbol's state.

Each symbol is identified by a unique *Symbol ID* value. When the IG receives a **Symbol Control** packet, the data in the packet shall be applied to the symbol corresponding to the specified symbol ID. If a symbol with that ID does not exist, then the IG shall ignore the packet. Each symbol shall be created independently with its own unique *Symbol ID* value even if two visually identical symbols are used.

Symbols may be attached to one another in a hierarchical relationship with each child symbol's position and rotation specified relative to its parent symbol's local coordinate system. The Host needs only to control the parent symbol's position and rotation in order to move all lower symbols in the hierarchy as a group. No explicit manipulation of a child symbol's position and rotation is necessary unless its position and rotation change with respect to its parent. Additionally, a child symbol may inherit certain display states from its parent.

The *Attach State* parameter of the **Symbol Control** packet determines whether a symbol is attached to a parent.

Figure 103 illustrates how symbols are able to be combined to form a hierarchy. Symbol 1 is a circle symbol arc centered at its local origin with a radius of three (3). Symbol 2 is a line symbol triangle defined with the same height as the arc's diameter. Symbol 3 is a text symbol with a font height of 2.4 and a center-left alignment. All symbols are initially hidden.

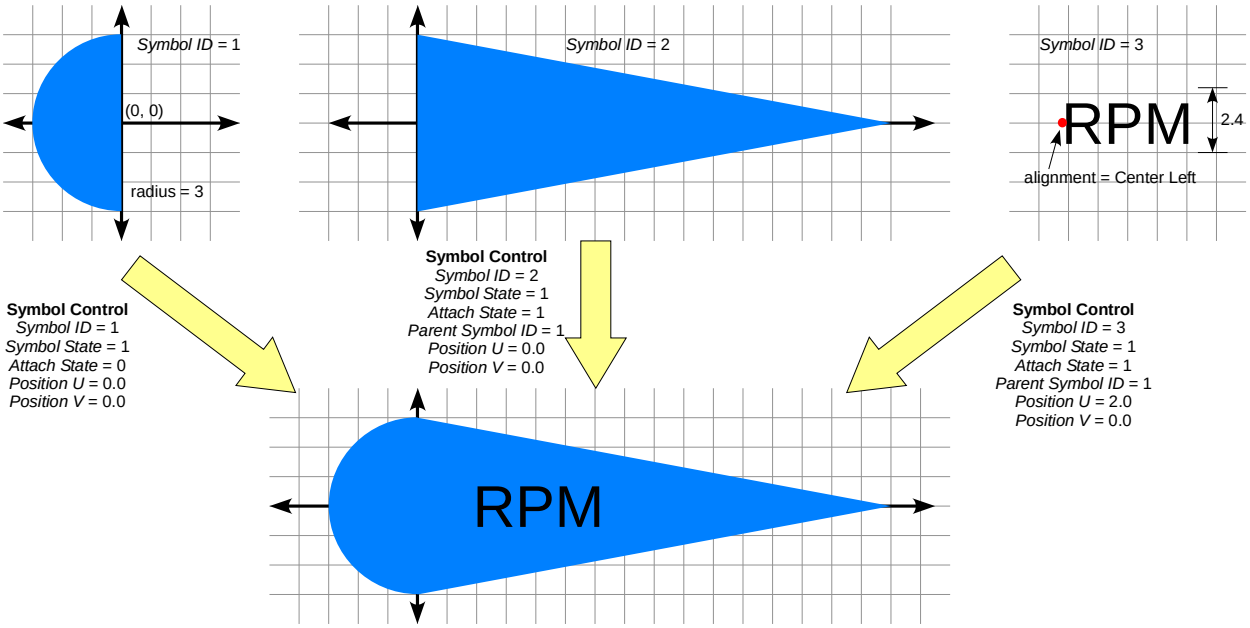


Figure 103 – Example of a Symbol Hierarchy

A **Symbol Control** packet is used to set the arc's *Symbol State* attribute to Visible (1). The arc is a top-level symbol so *Attach State* is set to Detach (0). A second **Symbol Control** packet sets the triangle's *Symbol State* to Visible (1) and to set its parent symbol as the arc. Finally, a third Symbol Control packet makes the text symbol visible, sets its parent symbol as the arc, and positions the text 2.0 units along the arc's **U** axis.

Note that the text has a higher *Symbol ID* value than the triangle, so the text is not occulted.

It is possible to rotate the entire hierarchy by rotating the circle symbol. As the circle symbol's local coordinate system rotates, all of its descendants are also rotated as shown in Figure 104:

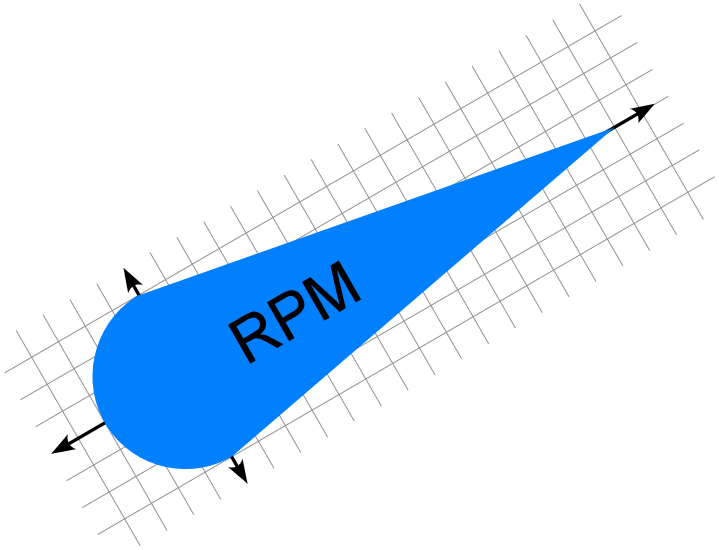


Figure 104 – Rotation of a Parent Symbol and its Children

The *Symbol State* field is used to control when a symbol is visible and when a symbol's geometry is unloaded. When a symbol is created, that symbol shall be hidden until the Host sends a **Symbol Control** packet with the *Symbol State* field set to Visible (1). Any immediate children of that symbol shall either remain hidden or become visible depending upon their individual states. The symbol and all of its children shall be hidden at any time by setting *Symbol State* to Hidden (0). When the Symbol is no longer needed, the Host sets the *Symbol State* to Destroyed (2) and the IG shall then unload the symbol and free any associated resources. Any children attached to the symbol shall also be destroyed.

The *Red*, *Blue*, *Green*, and *Alpha* parameters shall determine the color and transparency of a symbol. Alternatively, child symbols may inherit these values directly from their parents. If the *Inherit Color* parameter is set to Inherited (1), then the *Red*, *Blue*, *Green*, and *Alpha* parameters shall be ignored and the values of the parent shall be used. The *Inherit Color* parameter shall be ignored for top-level (i.e.,, root) symbols.

A symbol may flash or blink as determined by the *Flash Period* and *Flash Duty Cycle Percentage* parameters. The *Flash Period* parameter specifies the amount of time that shall elapse between two consecutive flashes. The *Flash Duty Cycle Percentage* parameter specifies the percentage of each flash cycle that the symbol shall be visible.

Figure 105 illustrates a typical flash period and duty cycle. The period is 0.4 seconds and the duty cycle is 75%. The symbol shall be visible for 0.3 seconds and is then invisible for 0.1 seconds. The cycle shall repeat itself until it is restarted or the symbol is destroyed.

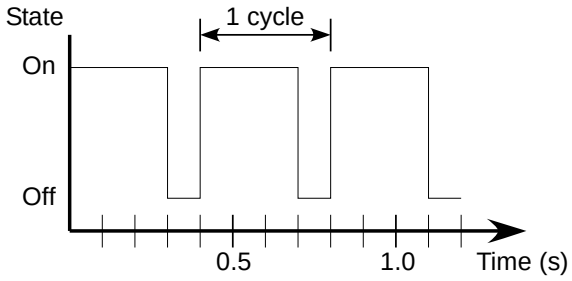


Figure 105 – Example of a Symbol Flash Cycle

If the *Flash Duty Cycle Percentage* parameter is set to 100%, then no flashing shall occur and the symbol shall always be visible.

If a symbol's duty cycle is less than 100%, then any descendents (child symbols, grandchildren, etc.) shall inherit the symbol's duty cycle and flash period. The *Flash Duty Cycle Percentage* and *Flash Period* attributes of the descendents shall be ignored. If a symbol flashes, then any descendents shall flash in synchronization with that symbol.

If a symbol's flash period or duty cycle is changed, then that symbol's flash cycle shall be restarted.

A symbol may be moved without resetting its flash cycle. If a **Symbol Control** packet's *Flash Control* parameter is set to Continue (0), and if the *Flash Period* and *Flash Duty Cycle Percentage* parameters have not changed for the given symbol, then the symbol's flash cycle shall not be reset. If the *Flash Control* parameter is set to Reset (1), then the symbol's flash cycle shall be restarted from the beginning.

The drawing order of symbols shall be determined by layer. Each symbol surface has 256 logical layers, which shall be rendered in order of increasing layer number. Symbols assigned to Layer 0 shall be drawn first, followed by the symbols on Layer 1, then those on Layer 2, etc. Symbols on higher-numbered layers may occult those on any lower-numbered layers. Symbols assigned to the same layer shall be drawn in order of increasing symbol ID.

The position of a top-level (root) symbol is specified with respect to the surface's 2D coordinate system (see Section 5.4.5.1). The position of a child symbol is specified with respect to the parent symbol's local coordinate system (see Section 5.4.5.2).

The *Scale U* and *Scale V* parameters specify scaling factors that the IG shall apply to a symbol's **U** and **V** dimensions, respectively. A ancestors' scaling factors shall also affect the symbol's apparent size as described in Section 5.4.5.2.

Once a **Symbol Control** packet is sent to the IG, the state of the specified symbol shall not change again until another **Symbol Control** or **Short Symbol Control** packet containing the same *Symbol ID* value is received, or until the symbol is implicitly deleted.

The contents of the **Symbol Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 48																*Packet ID* = 021h															
Symbol ID																*Parent Symbol ID*															
*5			*4	*3	*2	*1	*Layer*									*Flash Duty Cycle Percentage*								Reserved							
Surface ID																Reserved															
Flash Period																															
Position U																															
Position V																															
Rotation																															
Red								*Green*								*Blue*								*Alpha*							
Scale U																															
Scale V																															
Reserved																															

*1 *Symbol State*
*2 *Attach State*
*3 *Flash Control*
*4 *Inherit Color*
*5 Reserved

Figure 106 – Symbol Control Packet Structure

Table 41 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 41 – Symbol Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 48	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 48.
Packet ID Type: unsigned int16 Units: N/A Value: 021h	This parameter identifies this data packet as the **Symbol Control** packet. The Host shall set this parameter to 021h.
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol to which this packet shall be applied. This value shall be unique for each active symbol.

Parameter	Description
Parent Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the parent to which the IG shall attach the symbol. The Host may change this value without first detaching the symbol from its existing parent. If the specified parent symbol is invalid, then the IG shall not change the current attachment. If *Attach State* is set to Detach (0), then the IG shall ignore this parameter.
Symbol State Type: 2-bit field Units: N/A Values: 0 Hidden 1 Visible 2 Destroyed	This parameter specifies whether the symbol may be hidden, visible, or destroyed. The Host shall set this parameter to one of the following values: **Hidden** – The symbol shall be or remain hidden from view; however, the symbol may still be positioned, rotated, and scaled. It may also be attached to another symbol as a child. The symbol may also be used as a parent by other symbols, although any children shall also be hidden. **Visible** – The symbol shall be drawn on the surface. The symbol may be positioned, rotated, and scaled. It may also be attached to another symbol as a child. The symbol may also be used as a parent by other symbols. **Destroyed** – The symbol shall be deleted and any system resources should be released. Any children shall be destroyed. All other parameters in this packet shall be ignored. If this parameter is not set to one of the values listed above, then the IG shall ignore the packet. Note: Although the **Symbol Control** packet supports destruction of symbols, it is recommended that the **Short Symbol Control** packet be used for this purpose since all other parameters shall be ignored.

Parameter	Description
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the symbol may be attached as a child to a parent symbol. If this parameter is set to Detach (0), then the IG shall promote the symbol to, or leave it as, a top-level (non-child) symbol and shall ignore the *Parent Symbol* parameter. The IG shall place and orient the symbol according to the *U Position*, *V Position*, and *Rotation* parameters. Position and rotation shall be relative to the symbol surface's local coordinate system (see Section 5.4.5.1). If this parameter is set to Attach (1), then the IG shall attach or leave attached the symbol to that specified by the *Parent Symbol ID* parameter. The IG shall place and orient the symbol according to the *U Position*, *V Position*, and *Rotation* parameters. Position and rotation shall be relative to the parent symbol's local coordinate system (see Section 5.4.5.2). The attach state of a symbol may be changed at any time. The attachment or detachment shall take place immediately and shall remain in effect until changed with another **Symbol Control** packet or **Short Symbol Control** packet.
Flash Control Type: 1-bit field Units: N/A Values: 0 Continue 1 Reset	This parameter specifies whether the IG shall continue the symbol's flash cycle from its present state or restart it from the beginning. If either *Flash Duty Cycle Percentage* or *Flash Period* have changed, then the IG shall ignore this parameter and shall start the new flash sequence from the beginning. The IG may also ignore this parameter if *Flash Duty Cycle Percentage* is set to 0 or 100.
Inherit Color Type: 1-bit field Units: N/A Values: 0 Not Inherited 1 Inherited	This parameter specifies whether this symbol inherits its color from the symbol to which it is attached. If color is inherited, then this symbol's color, including the alpha component, shall be identical to the current color of the parent symbol. Note that the current color of the parent symbol may be inherited from another symbol. If *Attach State* is set to Detach (0), then the IG shall ignore this parameter.

Parameter	Description
Layer Type: unsigned int8 Units: N/A Values: 0 – 255	This parameter specifies the layer to which the symbol shall be assigned. The IG shall draw layers in order of increasing layer number. For example, Layer 0 shall be drawn first, followed by Layer 1, etc. Symbols on higher-numbered layers may occlude symbols on lower-numbered layers. If two or more symbols occupy the same layer, then the IG should draw the symbols in order of increasing symbol ID. Note that symbol layering is independent of the symbol transformation hierarchy. Any two siblings in a symbol hierarchy may or may not be assigned to the same layer or to adjacent layers.
Flash Duty Cycle Percentage Type: unsigned int8 Units: percent Values: 0 – 100	This parameter specifies the duty cycle for a flashing symbol. This period shall be measured as a percentage of the flash period. The IG shall make the symbol visible for the duration of the duty cycle period. If this value is set to zero (0), then the symbol shall always be invisible. If this value is set to 100%, then the symbol shall always be visible. The IG shall ignore this parameter if this symbol inherits its flashing behavior from its parent.
Surface ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol surface on which the IG shall draw the symbol. If the symbol is a child, then the IG shall use the top-level parent symbol's surface and shall ignore this parameter. If the specified surface is invalid and this symbol is not a child, then the IG shall not change the current attachment.
Flash Period Type: single float Units: seconds	This parameter specifies the duration of a single flash cycle. The IG shall make the symbol visible for the duration of the duty cycle period and invisible for the remainder of the flash period. The IG shall restart the flash cycle after each flash period. The IG shall ignore this parameter if *Flash Duty Cycle Percentage* is set to 0% or 100%. The IG shall ignore this parameter if the symbol inherits its flashing behavior from its parent.

Parameter	Description
Position U Type: single float Units: Scaled symbol surface units Datum: If *Attach State* = 0: Symbol surface's local 2D coordinate system If *Attach State* = 1: Parent symbol's local coordinate system	This parameter specifies the U component of the symbol's position. For top-level (non-child) symbols, the IG shall position the symbol with respect to the symbol surface's 2D coordinate system as described in Section 5.4.5.1. For child symbols, the IG shall position the symbol with respect to the parent symbol's local coordinate system as described in Section 5.4.5.2.
Position V Type: single float Units: Scaled symbol surface units Datum: If *Attach State* = 0: Symbol surface's local 2D coordinate system If *Attach State* = 1: Parent symbol's local coordinate system	This parameter specifies the V component of the symbol's position. For top-level (non-child) symbols, the IG shall position the symbol with respect to the symbol surface's 2D coordinate system as described in Section 5.4.5.1. For child symbols, the IG shall position the symbol with respect to the parent symbol's local coordinate system as described in Section 5.4.5.2.
Rotation Type: single float Units: degrees Values: 0 – 360 Datum: If *Attach State* = 0: Symbol surface's local 2D coordinate system If *Attach State* = 1: Parent symbol's local coordinate system	This parameter specifies the orientation of the symbol. The IG shall rotate the symbol counter-clockwise about the symbol's local origin. For top-level (non-child) symbols, the IG shall orient the symbol with respect to the symbol surface's local 2D coordinate system as described in Section 5.4.5.1. For child symbols, the IG shall orient the symbol with respect to the parent symbol's local coordinate system as described in Section 5.4.5.2.

Parameter	Description
Red Type: unsigned int8 Units: N/A	This parameter specifies the red component of the symbol's color. If *Inherit* Color is set to Not Inherited (0), then the IG shall draw each of the symbol's primitives using the color specified by this packet. If *Inherit Color* is set to inherit (1), then the IG shall ignore this parameter.
Green Type: unsigned int8 Units: N/A	This parameter specifies the green component of the symbol's color. If *Inherit* Color is set to Not Inherited (0), then the IG shall draw each of the symbol's primitives using the color specified by this packet. If *Inherit Color* is set to inherit (1), then the IG shall ignore this parameter.
Blue Type: unsigned int8 Units: N/A	This parameter specifies the blue component of the symbol's color. If *Inherit* Color is set to Not Inherited (0), then the IG shall draw each of the symbol's primitives using the color specified by this packet. If *Inherit Color* is set to inherit (1), then the IG shall ignore this parameter.
Alpha Type: unsigned int8 Units: N/A	This parameter specifies the alpha component of the symbol's color. If this parameter is set to zero (0), then the IG shall make the symbol fully transparent. If this parameter is set to 255, then the IG shall make the symbol fully opaque. If *Inherit* Color is set to Not Inherited (0), then the IG shall draw each of the symbol's primitives using the color specified by this packet. If *Inherit Color* is set to inherit (1), then the IG shall ignore this parameter.
Scale U Type: single float Units: N/A Values: > 0.0	This parameter specifies the scaling factor of the symbol along the symbol's local **U** axis. The IG shall apply this multiplier to the symbol's local coordinate system, which affects the symbol's geometry and any child symbols, as described in Section 5.4.5.2.

Parameter	Description
Scale V Type: single float Units: N/A Values: > 0.0	This parameter specifies the scaling factor of the symbol along the symbol's local **V** axis. The IG shall apply this multiplier to the symbol's local coordinate system, which affects the symbol's geometry and any child symbols, as described in Section 5.4.5.2.

6.1.35 Short Symbol Control

The **Short Symbol Control** packet is provided as a lower-bandwidth alternative to the **Symbol Control** packet (Section 6.1.34). The short control packet may be used when manipulation of only one or two symbol attributes of a symbol is necessary.

This packet allows for up to two symbol attributes to be modified. The attributes are specified by the *Attribute Select 1* and *Attribute Select 2* parameters. The values of these parameters shall determine what data types are used to interpret the *Attribute Value 1* and *Attribute Value 2* parameters, respectively.

Before the Host sends a **Short Symbol Control** referencing a symbol, the Host shall first send a **Symbol Control** packet referencing that symbol so all of the symbol's attributes are configured. The IG shall ignore any Short Symbol Control Packet that references a symbol whose state has not yet been fully established via a Symbol Control packet.

The contents of the **Short Symbol Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 24</td><td colspan="16">Packet ID = 022h</td></tr>
<tr><td colspan="16">Symbol ID</td><td colspan="8">Attribute Select 1</td><td colspan="8">Attribute Select 2</td></tr>
<tr><td colspan="2">*5</td><td colspan="2">*4</td><td>*3</td><td>*2</td><td colspan="2">*1</td><td colspan="24">Reserved</td></tr>
<tr><td colspan="32">Attribute Value 1</td></tr>
<tr><td colspan="32">Attribute Value 2</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *Symbol State*
*2 *Attach State*
*3 *Flash Control*
*4 *Inherit Color*
*5 Reserved

Figure 107 – Short Symbol Control Packet Structure

Table 42 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 42 – Short Symbol Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 24	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 022h	This parameter identifies this data packet as the **Short Symbol Control** packet. The Host shall set this parameter to 022h.

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol to which this packet shall be applied. This value shall be unique for each active symbol.
Attribute Select 1 Type: 8-bit field Units: N/A Values: 0 None 1 Surface ID 2 Parent Symbol ID 3 Layer 4 Flash Duty Cycle Percentage 5 Flash Period 6 Position U 7 Position V 8 Rotation 9 Color 10 Scale U 11 Scale V	This parameter identifies the attribute whose value is specified in the *Attribute Value 1* field. If this parameter is set to None (0), then *Attribute Value 1* shall be ignored.
Attribute Select 2 Type: 8-bit field Units: N/A Values: 0 None 1 Surface ID 2 Parent Symbol ID 3 Layer 4 Flash Duty Cycle Percentage 5 Flash Period 6 Position U 7 Position V 8 Rotation 9 Color 10 Scale U 11 Scale V	This parameter identifies the attribute whose value is specified in the *Attribute Value 2* field. If this parameter is set to None (0), then *Attribute Value 2* shall be ignored.

Parameter	Description
Symbol State Type: 2-bit field Units: N/A Values: 0 Hidden 1 Visible 2 Destroyed	This parameter specifies whether the symbol should be hidden, visible, or destroyed. The Host shall set this parameter to one of the following values: **Hidden** – The symbol shall be or remain hidden from view; however, the symbol may still be positioned, rotated, and scaled. It may also be attached to another symbol as a child. The symbol may also be used as a parent by other symbols, although any children shall also be hidden. **Visible** – The symbol shall be drawn on the surface. The symbol may be positioned, rotated, and scaled. It may also be attached to another symbol as a child. The symbol may also be used as a parent by other symbols. **Destroyed** – The symbol shall be deleted and any system resources should be released. Any children shall be destroyed. All other parameters in this packet shall be ignored. If this parameter is not set to one of the values listed above, then the IG shall ignore the packet. Note: Although the **Symbol Control** packet supports destruction of symbols, it is recommended that the **Short Symbol Control** packet be used for this purpose since all other parameters shall be ignored.

Parameter	Description
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the symbol may be attached as a child to a parent symbol. If this parameter is set to Detach (0), then the IG shall promote the symbol to, or leave it as, a top-level (non-child) symbol and shall ignore the *Parent Symbol* parameter. The IG shall place and orient the symbol according to the *U Position*, *V Position*, and *Rotation* parameters. Position and rotation shall be relative to the symbol surface's local coordinate system (see Section 5.4.5.1). If this parameter is set to Attach (1), then the IG shall attach or leave attached the symbol to that specified by the *Parent Symbol ID* parameter. The IG shall place and orient the symbol according to the *U Position*, *V Position*, and *Rotation* parameters. Position and rotation shall be relative to the parent symbol's local coordinate system (see Section 5.4.5.2). The attach state of a symbol may be changed at any time. The attachment or detachment shall take place immediately and shall remain in effect until changed with another **Symbol Control** packet or **Short Symbol Control** packet.
Flash Control Type: 1-bit field Units: N/A Values: 0 Continue 1 Reset	This parameter specifies whether the IG shall continue the symbol's flash cycle from its present state or restart it from the beginning. If either *Flash Duty Cycle Percentage* or *Flash Period* have changed, then the IG shall ignore this parameter and shall start the new flash sequence from the beginning. The IG may also ignore this parameter if *Flash Duty Cycle Percentage* is set to 0 or 100.
Inherit Color Type: 1-bit field Units: N/A Values: 0 Not Inherited 1 Inherited	This parameter specifies whether this symbol inherits its color from the symbol to which it is attached. If color is inherited, then this symbol's color, including the alpha component, shall be identical to the current color of the parent symbol. Note that the current color of the parent symbol may be inherited from another symbol. If *Attach State* is set to Detach (0), then the IG shall ignore this parameter.

Parameter	Description
Attribute Value 1 Type: Attribute-specific Units: Attribute-specific	This parameter specifies the value of the attribute identified by the *Attribute Select 1* field. If *Attribute Select 1* is set to Surface ID (1), Parent Symbol ID (2), Layer (3), or Flash Duty Cycle Percentage (4), then *Attribute Value 1* shall be treated as a 32-bit integer. If *Attribute Select 1* is set to Flash Period (5), Position U (6), Position V (7), Rotation (8), Scale V (10), or Scale V (11), then *Attribute Value 1* shall be treated as a 32-bit single-precision floating-point number. If *Attribute Select 1* is Color (9), then *Attribute Value 1* shall be treated as four 8-bit integers specifying each of the four color components. The most significant byte specifies the red component, followed by the green component, then blue, and finally alpha. Regardless of the attribute, the IG shall byte-swap this parameter as a 32-bit value if byte-swapping is required.
Attribute Value 2 Type: Attribute-specific Units: Attribute-specific	This parameter specifies the value of the attribute identified by the *Attribute Select 2* field. If *Attribute Select 2* is set to Surface ID (1), Parent Symbol ID (2), Layer (3), or Flash Duty Cycle Percentage (4), then *Attribute Value 2* shall be treated as a 32-bit integer. If *Attribute Select 2* is set to Flash Period (5), Position U (6), Position V (7), Rotation (8), Scale V (10), or Scale V (11), then *Attribute Value 2* shall be treated as a 32-bit single-precision floating-point number. If *Attribute Select 2* is Color (9), then *Attribute Value 2* shall be treated as four 8-bit integers specifying each of the four color components. The most significant byte specifies the red component, followed by the green component, then blue, and finally alpha. Regardless of the attribute, the IG shall byte-swap this parameter as a 32-bit value if byte-swapping is required.

6.1.36 Symbol Textured Circle Definition

The **Symbol Textured Circle Definition** packet is used to create a texture mapped symbol circle surface and control the circle's radius, start and end angles, texture position relative to the center point, rotation, wrapping (repeat versus clamp), and filtering (nearest versus linear), and other attributes. Only filled circles can be defined with this packet. Outline circles are limited to the **Symbol Circle Definition** packet.

Each circle symbol is identified by a *Symbol ID* value that is unique from all other symbols (including text and circle symbols). Every symbol shall be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

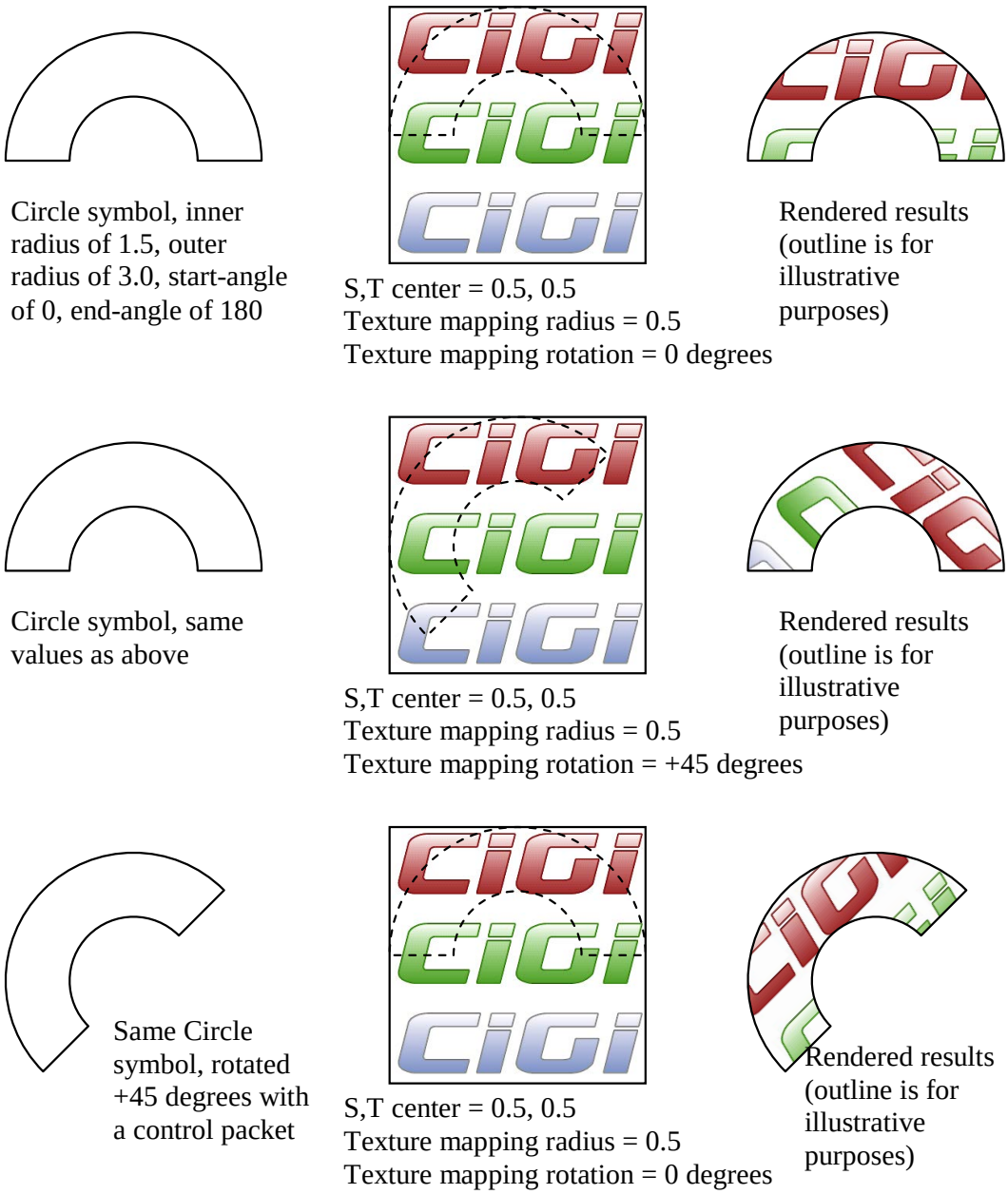


Figure 108 – Example of Texture Mappings on a Symbol Textured Circle

Once a **Symbol Textured Circle Definition** packet describing a circle symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Circle Definition**, **Symbol Polygon Definition**, **Symbol Textured Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

If the texture applied to the symbol contains an alpha channel, then the symbol's appearance shall be modulated by the texel alpha in combination with the usual symbol alpha values. IG vendors are encouraged to set up their texture blending parameters (for example, glAlphaFunc() in OpenGL parlance) in order to achieve smooth blending for symbols with alpha-enabled textures.

The texture coordinates are to be interpreted such that S=0, T=0 shall correspond to the lower left corner of the texture and S=1, T=1 corresponds to the upper right corner. The S,T center-point coordinate shall determine where the center point of the circle is mapped in normalized S,T space (see top of Figure 108). The texture-mapping rotation shall rotate the texture with respect to the circle (see middle of Figure 108).

Texturing is applied before any symbol rotation. If a symbol is rotated via a Symbol Control, and no other changes are made to the symbol, then the texture shall rotate with the symbol (see bottom of Figure 108). The same is true of other attributes set by Symbol Control packets, such as the symbol scaling and translation values.

The texture-mapping radius determines the extents of the texture that is stretched to cover the circle. The circle's texture mapping coordinates are scaled such that the distance, in normalized S,T space (i.e., not symbol units), between the circle's center and its outer radius corresponds to this value. Smaller values here shall have a "magnifying" effect when the texture is applied to the circle, because a smaller region of texture is being stretched to cover the same amount of symbol area. Larger values have the opposite effect, for the same reason.

Textures should be installed on the IG, similar to entity models.

The contents of the **Symbol Textured Circle Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet size= 16 + (40 x n)																*Packet ID = 023h*															
Symbol ID																*Texture ID*															
Reserved								*2								*1								Reserved							
Reserved																															
Center U																															
Center V																															
Radius																															
Inner Radius																															
Start Angle																															
End Angle																															
Texture Coordinate S at Center Point																															
Texture Coordinate T at Center Point																															
Texture-mapping Radius																															
Texture-mapping Rotation																															
•																															
•																															
•																															
Center U																															
Center V																															
Radius																															
Inner Radius																															
Start Angle																															
End Angle																															
Texture Coordinate S at Center Point																															
Texture Coordinate T at Center Point																															
Texture-mapping Radius																															
Texture-mapping Rotation																															

*1 - Texture filter mode
*2 - Texture repeat or clamp

Figure 109 – Symbol Textured Circle Definition Packet Structure

Table 43 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 43 – Symbol Textured Circle Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: $16 \le [16 + (40 \times n)] \le 65496$ where n is the number of circles described by this packet	This parameter specifies the number of bytes in this data packet. The Host shall set this size using the formula in the left column. This value shall be divisible by eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 023h	This parameter identifies this data packet as the **Symbol Textured Circle Definition** packet. The IG shall set this parameter to 023h.
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a textured circle symbol with the specified identifier already exists, then the IG shall replace the existing symbol geometry and shall retain the existing state. If a symbol with the specified identifier already exists, and if that symbol is of a type other than a textured circle, then the IG shall destroy the existing symbol and any children and shall create a new symbol. Note that the Host shall subsequently send a **Symbol Control** packet to set the new symbol's visibility and state.
Texture ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the texture that is being applied. This identifier shall be unique among all existing textures.

Parameter	Description
Texture Filter Mode Type: 1-bit field Units: N/A Values: 0 Nearest 1 Linear	This parameter specifies the type of texture filtering/interpolation applied to the symbol. If the Host sets this parameter to Nearest (0), then the IG shall not smooth the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_MIN_FILTER and GL_TEXTURE_MAG_FILTER to GL_NEAREST. If the Host sets this parameter to Linear (1), then the IG shall smooth the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_MIN_FILTER to GL_LINEAR_MIPMAP_LINEAR and setting GL_TEXTURE_MAG_FILTER to GL_LINEAR.
Texture repeat or clamp Type: 1-bit field Units: N/A Values: 0 Repeat 1 Clamp	This parameter specifies whether texture coordinates are to be wrapped or clamped when applied to the symbol. If the Host sets this parameter to Repeat (0), then the IG shall repeat the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T to GL_REPEAT. If the Host sets this parameter to Clamp (1), then the IG shall not repeat the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T to GL_CLAMP_TO_EDGE.
Center U Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the u position of the circle's center. The IG shall place the center with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Center V Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the v position of the circle's center. The IG shall place the center with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).

Parameter	Description
Radius Type: single float Units: Scaled symbol surface units Values: ≥ 0.0 Datum: Center of circle	This parameter specifies the radius of the outer circumference of the circle or arc. This value shall be measured in scaled symbol surface units (see Section 5.4.5.2). The IG shall create a filled region bounded by the inner and outer radii and the start and end angles as illustrated in Figure 98.
Inner Radius Type: single float Units: Scaled symbol surface units Value: ≥ 0.0 to < *Radius* Datum: Center of circle	This parameter specifies the inner radius of a filled circle or arc. This value shall be measured in scaled symbol surface units (see Section 5.4.5.2). The IG shall create a filled region bounded by the inner and outer radii and the start and end angles.
Start Angle Type: single float Units: degrees Value: 0.0 – 360.0 Datum: Symbol's +**U** axis	This parameter specifies the starting angle of the arc. This angle shall be measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc shall cross the +**U** axis.
End Angle Type: single float Units: Scaled symbol surface units Value: 0.0 – 360.0 Datum: Symbol's +**U** axis	This parameter specifies the ending angle of the arc. This angle shall be measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc shall cross the +**U** axis.
Texture Coordinate S at Center Point Type: single float Units: Normalized Value: 0.0 – 1.0 Datum: Texture coordinate system	This parameter specifies the normalized S texture coordinate to position the texture relative to the circle's center. The texture coordinates shall be interpreted such that *S*=0, *T*=0 corresponds to the lower left corner of the texture, and *S*=1, *T*=1 corresponds to the upper right corner.

Parameter	Description
Texture Coordinate T at Center Point Type: single float Units: Normalized Value: 0.0 – 1.0 Datum: Texture coordinate system	This parameter specifies the normalized T texture coordinate to position the texture relative to the circle's center. The texture coordinates shall be interpreted such that *S*=0, *T*=0 corresponds to the lower left corner of the texture, and *S*=1, *T*=1 corresponds to the upper right corner.
Texture-mapping Radius Type: single float Units: Normalized Datum: Texture coordinate system	This parameter specifies the extents of the texture that is stretched to cover the circle. The distance in S,T space between the circle's center and its outer radius shall correspond to unity. Smaller values shall have a "magnifying" effect and larger values shall have the opposite effect.
Texture-mapping Rotation Type: single float Units: degrees Datum: Texture Symbol's +**U** axis	This parameter specifies the rotation of the texture relative to the circle's center. Positive values shall shift the circle's S,T mapping counter-clockwise and negative values shall shift it clockwise.

6.1.37 Symbol Textured Polygon Definition

The **Symbol Textured Polygon Definition** packet is used to create a texture mapped symbol polygon surface and control the texture position relative to the vertices, wrapping (repeat versus clamp), and filtering (nearest versus linear). Only filled geometry, such as Triangles, Triangle Strips, and Triangle Fans can be defined with this packet. The parameter **Primitive Type** shall not be set to Point, Line, Line Strip, or Line Loop.

Each polygon symbol is identified by a *Symbol ID* value that is unique from all other symbols (including text and circle symbols). Every symbol shall be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

Once a **Symbol Textured Polygon Definition** packet describing a polygon symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Circle Definition**, **Symbol Textured Circle Definition**, **Symbol Polygon Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle symbol shall be destroyed along with any children and a new symbol shall be created using the new definition packet.

If the texture applied to the symbol contains an alpha channel, then the symbol's appearance shall be modulated by the texel alpha in combination with the usual symbol alpha values. IG vendors are encouraged to set up their texture blending parameters (for example, glAlphaFunc() in OpenGL parlance) in order to achieve smooth blending for symbols with alpha-enabled textures.

Every polygon symbol is defined as an ordered set of zero or more points. Each point shall be defined with respect to the symbol's 2D coordinate system (see Section 5.4.5.2) by a pair of coordinates specified in the *Vertex U* and *Vertex V* parameters.

The method and order by which the points are connected is determined by the *Primitive Type* parameter. These primitives are described in Table 38.

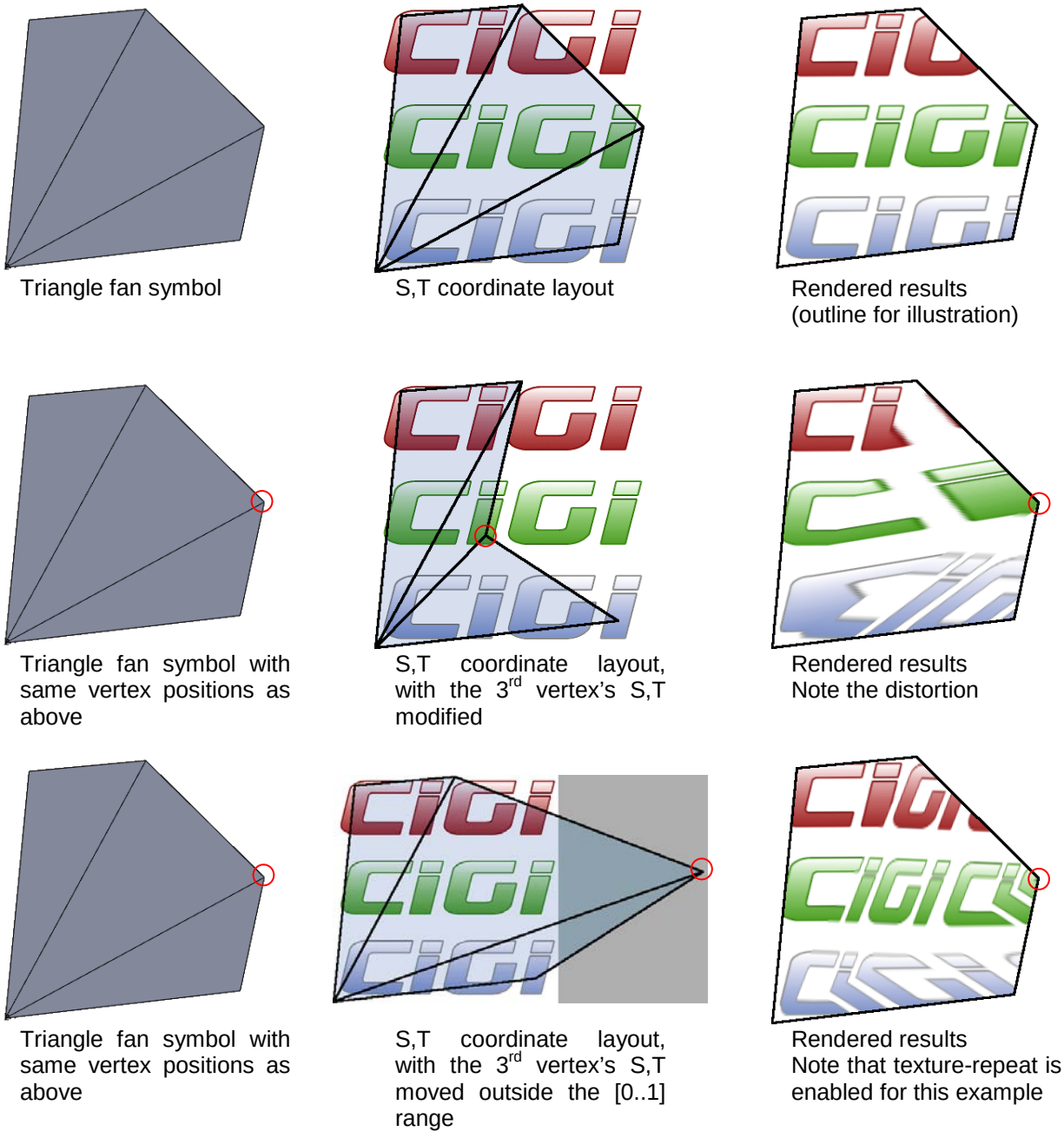


Figure 110 – Example of Texture Mappings on a Symbol Textured Polygon

The texture coordinates are to be interpreted such that S=0, T=0 shall correspond to the lower left corner of the texture and S=1, T=1 corresponds to the upper right corner (see top of Figure 110). The per-vertex S and T values are applied in a manner that should be familiar to users of OpenGL or Direct3D. Figure 110 shows some sample use-cases.

Textures should be installed on the IG, similar to 3D entity models.

The contents of the **Symbol Textured Polygon Definition** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet size = 16 + (16 x n)																*Packet ID= 024h*															
Symbol ID																*Texture ID*															
Primitive Type				**3*				**2*				**1*				Reserved															
Reserved																															
Vertex U_1																															
Vertex V_1																															
Texture Coordinate S																															
Texture Coordinate T																															
•																															
•																															
•																															
Vertex U_n																															
Vertex V_n																															
Texture Coordinate S																															
Texture Coordinate T																															

*1 - Texture filter mode
*2 - Texture repeat or clamp
*3 - Reserved

Figure 111 – Symbol Textured Polygon Definition Packet Structure

Table 44 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 44 - Symbol Textured Polygon Definition Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: $16 \le [16 + (40 \times n)] \le 65496$ where n is the number of vertices described by this packet	This parameter specifies the number of bytes in this data packet. The Host shall set this size using the formula in the left column. This value shall be divisible by eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 024h	This parameter identifies this data packet as the **Symbol Textured Polygon Definition** packet. The IG shall set this parameter to 024h.

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier shall be unique among all existing symbols. If a textured line symbol with the specified identifier already exists, then the IG shall replace the existing symbol geometry and shall retain the existing state. If a symbol with the specified identifier already exists, and if that symbol is of a type other than a textured line, then the IG shall destroy the existing symbol and any children and shall create a new symbol. Note that the Host shall subsequently send a **Symbol Control** packet to set the new symbol's visibility and state.
Texture ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the texture that is being applied. This identifier shall be unique among all existing textures.
Primitive Type Type: 4-bit field Units: N/A Values: *0 - 3* *not valid* 4 Triangle 5 Triangle Strip 6 Triangle Fan	This parameter specifies the type of point or line primitive defined by this packet. Each primitive type is described in Table 38. The IG shall ignore packets specifying values 0 – 3.
Texture Filter Mode Type: 1-bit field Units: N/A Values: 0 Nearest 1 Linear	This parameter specifies the type of texture filtering/interpolation applied to the symbol. If the Host sets this parameter to Nearest (0), then the IG shall not smooth the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_MIN_FILTER and GL_TEXTURE_MAG_FILTER to GL_NEAREST. If the Host sets this parameter to Linear (1), then the IG shall smooth the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_MIN_FILTER to GL_LINEAR_MIPMAP_LINEAR and setting GL_TEXTURE_MAG_FILTER to GL_LINEAR.

Parameter	Description
Texture repeat or clamp Type: 1-bit field Units: N/A Values: 0 Repeat 1 Clamp	This parameter specifies whether texture coordinates are to be wrapped or clamped when applied to the symbol. If the Host sets this parameter to Repeat (0), then the IG shall repeat the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T to GL_REPEAT. If the Host sets this parameter to Clamp (1), then the IG shall not repeat the texture applied to this Symbol. In OpenGL parlance have the effect of setting GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T to GL_CLAMP_TO_EDGE.
Vertex U Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the u position of a vertex. The IG shall place the vertex with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Vertex V Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the v position of a vertex. The IG shall place the center with respect to the symbol's local coordinate system. This position shall be measured in scaled symbol surface units (see Section 5.4.5.2).
Texture Coordinate S Type: single float Units: Normalized Value: 0.0 – 1.0 Datum: Texture coordinate system	This parameter specifies the normalized S texture coordinate to position the texture relative to the associated vertex. The texture coordinates shall be interpreted such that *S*=0, *T*=0 corresponds to the lower left corner of the texture, and *S*=1, *T*=1 corresponds to the upper right corner.
Texture Coordinate T Type: single float Units: Normalized Value: 0.0 – 1.0 Datum: Texture coordinate system	This parameter specifies the normalized T texture coordinate to position the texture relative to the associated vertex. The texture coordinates shall be interpreted such that *S*=0, *T*=0 corresponds to the lower left corner of the texture, and *S*=1, *T*=1 corresponds to the upper right corner.

6.1.38 Entity Control

The **Entity Control** packet is used to create and destroy entity instances, to establish a relationship between an *Entity ID* and the entity type to be created/destroyed, as well as to control attributes that describe an entity's state that are modified on an infrequent basis. Additional controls over the entity's behavior are described in Sections 6.1.2 **Entity Position**, 6.1.6 **Articulated Part Control**, and 6.1.39 **Animation Control**.

Each entity shall be identified by a unique identifier called the Entity ID. When the Host sends an **Entity Control** packet to the IG, the IG sets the state of the entity object corresponding to the value of the *Entity ID* parameter. If the specified entity does not exist, and if the specified entity type is valid, then the IG shall create the entity and display it per the *Entity State* described below.

When the IG creates an entity, it shall instantiate the model corresponding to the value of the entity type record. This instance shall exist as a unique and independent model hierarchy within the IG, therefore, any operations that modify an entity's model hierarchy (i.e., part articulations) shall only affect that entity and its children.

Figure 112 illustrates the assignment of unique Entity ID values to multiple entities of the same type. The number assignments in this example are hypothetical.

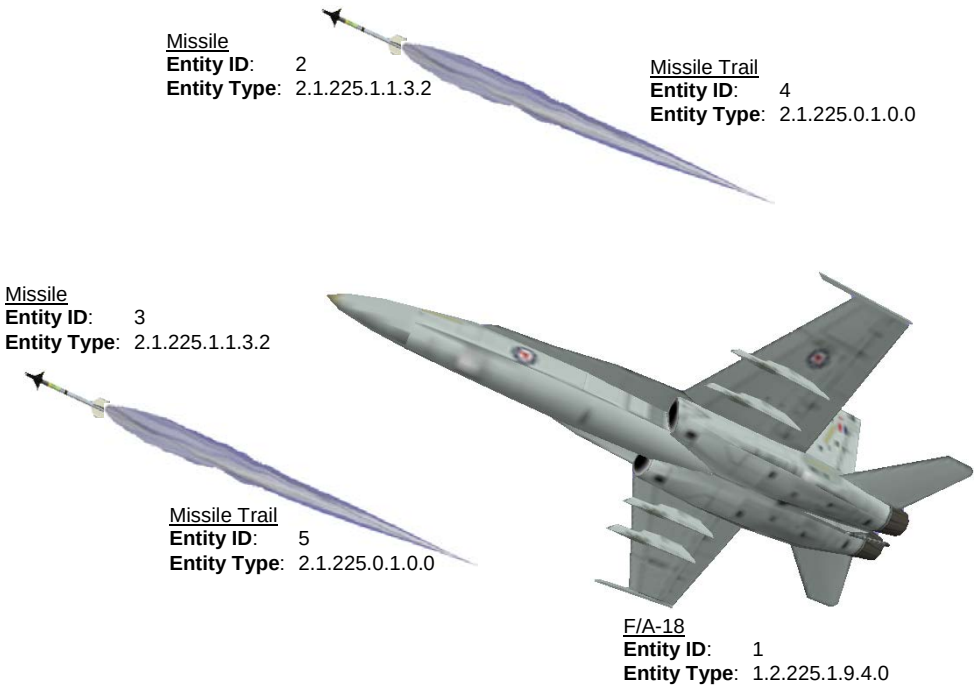


Figure 112 – Example of Entity Instantiation

The entity type is defined either as a 16-bit *Short Entity Type* or as an extended, 64-bit composite record made up of seven (7) parameters that are used to specify the kind of entity, the country of design, the domain, the specific identification of the entity, and any extra information necessary for describing the entity. The content and intended usage of the extended entity type record closely parallels the Entity Type record used by [DIS]. It is intended that [ESI] be used as a reference for defining the values for enumerations that shall be used to populate these parameters (16-bit values are available in the XML document under the 'uid' attribute). While these DIS-derived entity types have traditionally been used by military simulations, new entity types may be registered and are not limited to the military domain. The *Extended Entity Type* parameter defines whether a 16-bit or 64-bit entity type is to be used for the specified *Entity ID.* If the Host sets the entity type (any format) to zero (0), then the IG shall produce an

entity with no geometry. Such an entity might be used to represent an entity attached to an Out-the-Window view or an invisible spectator. If the specified entity type is undefined and *Entity Type Substitution Enable* in **IG Control** is Disabled (0), the IG shall not perform any entity type substitution.

When changing entity types, the Host shall first delete the entity by setting the *Entity State* parameter to Destroyed (2) and then recreate the entity and any children during a subsequent frame.

The *Entity State* parameter is used to control when an entity is visible and when its geometry is loaded and unloaded. If the *Entity State* is set to Inactive/Standby (0) when instancing a new entity, the IG shall load the geometry. The Host shall send an **Entity Position** packet with an initial position in the same frame in which the *Entity State* is set to Active (1) or in a prior frame. The IG should report an error if an entity is activated without an initial position and the entity shall remain Inactive (0). When the *Entity State* is set to Active (1), the entity shall be added to the scene. When the entity's state is transitioned from Active (1) to Inactive/Standby (0), the entity and any children shall be disabled and hidden from the scene. When the *Entity State* is set to Destroyed (2) the IG shall remove the model from its internal data structures representing the graphical scene. Any children attached to the entity shall also be destroyed.

Models may be preloaded to increase the initial display speed. For example, when an aircraft fires a missile, a missile entity will be used for that missile. Unless the missile geometry is cached, the IG needs to load the model from its hard disk. Because of its speed, the missile might fly a significant distance (and possibly beyond visual range) before the disk I/O can be completed. By preloading the entity, the geometry already exists in memory and is available for instant activation within the graphical scene when needed. To accomplish this, the Host could create the missile with an **Entity Control** packet with the *Entity State* parameter set to Inactive/Standby (0). When the missile is needed later, the Host would send an **Entity Control** packet for that entity with the *Entity State* parameter set to Active (1) and an **Entity Position** packet containing the proper positional data.

The opacity of an entity's geometry shall be controlled by the Alpha parameter. The *Inherit Alpha* parameter indicates whether a child entity's alpha value shall be combined with that of its parent. For example, a missile attached to the wing of an aircraft would typically be made invisible when the aircraft is destroyed, so its *Inherit Alpha* attribute would be set to Inherited (1). An explosion or similar animation attached to that aircraft, however, would typically linger after the aircraft's destruction, so its *Inherit Alpha* attribute would be set to Not Inherited (0).

Note that setting the *Entity State* parameter to Inactive/Standby is not equivalent to setting the *Alpha* parameter to zero (0).

The contents of the **Entity Control** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 16</td><td colspan="16">Packet ID = 025h</td></tr>
<tr><td colspan="2">Reserved</td><td>*5</td><td>*4</td><td>*3</td><td>*2</td><td>*1</td><td colspan="9">Alpha</td><td colspan="16">Entity ID</td></tr>
<tr><td colspan="8">Entity Kind</td><td colspan="8">Entity Domain</td><td colspan="16">Entity Country or Short Entity Type</td></tr>
<tr><td colspan="8">Entity Category</td><td colspan="8">Entity Subcategory</td><td colspan="8">Entity Specific</td><td colspan="8">Entity Extra</td></tr>
</table>

*1 *Entity State*
*2 *Collision Reporting Enable*
*3 *Inherit Alpha*
*4 *Smoothing Enable*
*5 *Extended Entity Type*

Figure 113 – Entity Control Packet Structure

Table 45 defines each parameter's data type, units, and usage.

Table 45 – Entity Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The Host shall set this parameter to 16.
Packet ID Type: unsigned int16 Units: N/A Value: 025h	This parameter identifies this data packet as the **Entity Control** packet. The Host shall set this parameter to 025h.
Entity State Type: unsigned 2-bit field Units: N/A Values: 0 Inactive/Standby 1 Active 2 Destroyed	The value of this parameter specifies whether the entity may be active, inactive, or destroyed. This parameter may be set to one of the following values: **Inactive/Standby** – The entity shall be invisible. Additionally, the entity shall be excluded from line of sight and collision testing. **Active** – The entity shall be visible and shall be included in line of sight and collision testing. **Destroyed** – The entity shall be invisible. Any children shall also be destroyed. All other parameters in this packet shall be ignored.

Parameter	Description
Collision Reporting Enable Type: 1-bit field Units: N/A Values: 0 Disabled 1 Enabled	The value of this parameter determines whether any collision detection segments and volumes associated with this entity are used as the source in collision testing. If the Host sets this parameter to Enabled (1), then each collision detection segment shall be tested every frame for intersections with polygons not associated with this entity and each collision detection volume shall be tested pair-wise with every other volume that is not associated with the entity. If the Host sets this parameter to Disabled (0), then any collision detection segments defined for the entity shall be ignored and any collision detection volumes shall only be tested (as the destination) against volumes defined for entities whose collision reporting is enabled. See Sections 6.1.22 and 6.1.23 for details on creating collision detection segments and volumes, respectively.
Inherit Alpha Type: 1-bit field Units: N/A Values: 0 Not Inherited 1 Inherited	This parameter specifies whether the entity's alpha shall be combined with the apparent alpha of its parent. **Not Inherited** – The entity's apparent alpha shall not be affected by the apparent alpha of the parent entity. **Inherited** – The entity's apparent alpha shall be directly proportional to the apparent alpha of the parent entity as described in the text accompanying the *Alpha* parameter. Note that a change in an entity's alpha shall affect the entities below it in the hierarchy if those entities inherit their parents' alphas. If *Attach State* is set to Detach (0), then the IG shall ignore this parameter.
Smoothing Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the IG shall smooth the entity's motion by extrapolation or interpolation. Such smoothing may be useful for compensating for lost CIGI messages, irregular frame rates, asynchronous operation, etc. If extrapolation or interpolation is disabled globally through the *Smoothing Enable* flag of the **IG Control** packet, then smoothing shall not be applied to the entity.

Parameter	Description
Extended Entity Type Type: 1-bit field Units: N/A Values: 0 Short 1 Extended	This parameter specifies whether the IG shall use a 16-bit *Short Entity Type* definition for the *Entity ID* or an extended 64-bit definition using parameters *Entity Country, Entity Domain, Entity Kind, Entity Extra, Entity Specific, Entity Subcategory, and Entity Category.* Note parameters *Short Entity Type* and *Entity Country* share the same packet field.
Alpha Type: unsigned int8 Units: N/A	This parameter specifies the explicit alpha that shall be applied to the entity's geometry. A value of zero (0) corresponds to full transparency. A value of 255 corresponds to full opacity. Intermediate values should be interpolated linearly. The *Inherit Alpha* parameter may further affect how this parameter is applied. If the *Inherit Alpha* parameter is set to Inherited (1) and if the entity is a child entity, then the apparent alpha shall be determined by the following formula: $$\alpha = \frac{\alpha_0 \alpha_p}{255}$$ where α is the apparent alpha of the child, α_0 is the explicit alpha of the child, and α_p is the apparent alpha of the parent. If the *Inherit Alpha* parameter is set to Not Inherited (0), or if the entity is a top-level entity, then the *Alpha* parameter shall be applied directly.
Entity ID Type: unsigned int16 Units: N/A	This parameter identifies a specific entity. The IG shall apply the values in this packet to the state of the entity corresponding to this value.
Entity Kind Type: unsigned int8 Units: N/A	This field shall identify the kind of entity described by the Entity Type record. This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].
Entity Domain Type: unsigned int8 Units: N/A	This field shall specify the domain in which the entity operates (i.e., subsurface, surface, land) except for munition entities. For munition entities this field shall specify the domain of the target (for example, the munition might be a surface-to-air, so the domain would be anti-air). This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].

Parameter	Description
Short Entity Type Type: unsigned int16 Units: N/A	If *Extended Entity Type* is Short (0), then this field shall be used for 16-bit *Short Entity Type* values, for example from CIGI 3.x or [ESI] 16-bit Unique Identification Numbers (UIDs). The other entity sub-type parameters shall be ignored.
Entity Country Type: unsigned int16 Units: N/A	If the *Extended Entity Type* is Extended (1), then this field shall specify the country to which the design of the entity is attributed. This field shall be represented by a 16-bit enumeration. Values for this field are found in [ESI].
Entity Category Type: unsigned int8 Units: N/A	This field shall specify the main category that describes the entity. This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].
Entity Subcategory Type: unsigned int8 Units: N/A	This field shall specify a particular subcategory to which an entity belongs based on the Category field. This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].
Entity Specific Type: unsigned int8 Units: N/A	This field shall specify specific information about an entity based on the Subcategory field. This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].
Entity Extra Type: unsigned int8 Units: N/A	This field shall specify extra information required to describe a particular entity. The contents of this field depends on the type of entity represented. This field shall be represented by an 8-bit enumeration. Values for this field are found in [ESI].

6.1.39 Animation Control

Animations are entity features that are played in a frame sequence mode like a movie or played in a programmed behavior mode like a particle system.

The *Animation State* parameter shall be used to control entity animations. The entity's animation functionality shall conform to the Animation State Summary table below.

Note that setting the *Animation State* parameter to Stop shall have different effects on different types of animations. Frame-based animations may simply stop, or begin a termination sequence if such a sequence has been defined, at the current frame. Emitter-based animations (e.g, missile trails and particle systems) shall stop producing new particles or segments; however, existing particles or segments shall continue to decay normally. Stopping an animation does not implicitly remove it from the scene unless the *Entity State* parameter is set to Inactive/Standby or Destroyed.

The following table summarizes the animation behavior:

Table 46 – Animation State Summary

Current State	New *Animation State* Value	New *Animation Frame Position Reset* Value	Effect
Stop	Stop	Continue	None
	Stop	Reset	None
	Play	Continue	Plays from beginning
	Play	Reset	Restarts from beginning
Play	Stop	Continue	Stops at current frame or stops emitting particles/segments; Switches to termination sequence or decay behavior if defined
	Stop	Reset	Same as *Continue*
	Play	Continue	No action, continues playing
	Play	Reset	Restarts from beginning
Animation Speed = 0	Stop	Continue	Stops at current frame; Stops emitting particles/segments
	Stop	Reset	Same as *Continue*
	Play	Continue	Pauses at current frame; Freezes all particles
	Play	Reset	Freezes all particles; Resets to beginning
Animation Speed > 0	Stop	Continue	Stops at current frame or stops emitting particles/segments; Switches to termination sequence or decay behavior if defined at specified speed
	Stop	Reset	Same as *Continue*
	Play	Continue	Play resumes at specified speed
	Play	Reset	Restarts from beginning at specified speed

If an animation has been built with a limited duration, and if the *Animation Loop Mode* parameter is set to One-Shot, then the animation shall stop automatically upon its completion. The IG shall report this event by sending an **Animation Stop Notification** packet (Section 6.2.15) to the Host. If the *Animation Loop Mode* parameter is set to Loop, the animation shall immediately restart from the beginning and no **Animation Stop Notification** packet shall be sent.

Note that a negated *Animation Speed* parameter value shall play the animation backward.

The contents of the **Animation Control** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 16																*Packet ID* = 026h															
Reserved				*4	*3	*2	*1	*Alpha*								*Entity ID*															
Animation ID																Reserved															
Animation Speed																															

*1 *Animation State*
*2 *Animation Frame Position Reset*
*3 *Animation Loop Mode*
*4 *Inherit Alpha*

Figure 114 – Animation Control Packet Structure

Table 47 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 47 – Animation Control Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter shall be 16.
Packet ID Type: unsigned int16 Units: N/A Value: 026h	This parameter identifies this data packet as the **Animation Control** packet. The value of this parameter shall be 026h.
Animation State Type: 1-bit field Units: N/A Values: 0 Stop 1 Play	This parameter specifies the state of an animation. **Stop** – Stops the animation sequence. If the animation has a termination sequence or decay behavior, the animation shall switch to that behavior. Has no effect if the animation is currently stopped. **Play** – The animation geometry is visible. Play behavior is affected by the *Animation Frame Reset, Animation Loop Mode, and Animation Speed* parameters

Parameter	Description
Animation Frame Position Reset Type: 1-bit field Units: N/A Values: 0 Continue 1 Reset	This parameter determines whether the animation is restarted from the beginning or just continues from its current frame. The value Reset (1) shall be ignored when *Animation State* is set to Stop (0).
Animation Loop Mode Type: 1-bit field Units: N/A Values: 0 One-Shot 1 Continuous	This parameter specifies whether an animation should be a one-shot (i.e.,, should play once and stop) or should loop continuously.
Inherit Alpha Type: 1-bit field Units: N/A Values: 0 Not Inherited 1 Inherited	This parameter specifies whether the animation's alpha is combined with the apparent alpha of the entity. The following formula shall be used to combine the alpha values: $$\alpha = \frac{\alpha_0 \alpha_p}{255}$$ Where α is the apparent alpha of the animation, α_0 is the explicit alpha of the animation, and α_p is the apparent alpha of the entity.
Alpha Type: unsigned int8 Units: N/A	This parameter specifies the explicit alpha to be applied to the entity's geometry. A value of zero (0) corresponds to fully transparent; a value of 255 corresponds to fully opaque.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the animation belongs.
Animation ID Type: unsigned int16 Units: N/A	This parameter specifies the animation to which the data in this packet should be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular animation within the simulation.
Animation Speed ***Type: float*** ***Units: N/A***	This parameter specifies the speed at which the animation is played. A positive speed is forward. A negative speed is backward. A speed of 0 is paused. A speed of 1.0 is the normal speed. This parameter only applies when the *Animation State* is Play.

6.2 IG-to-Host Packets

6.2.1 Start of Frame

The **Start of Frame** packet is used to signal the beginning of a new frame. Every IG-to-Host message shall contain *exactly one* **Start of Frame** packet. This packet shall be the *first* packet in the message.

The contents of the **Start of Frame** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = FFFFh															
Major Version = 4								*Database Number*								Reserved								*Minor Version*				*3	*2	*1	
IG Frame Number																															
Timestamp																															
Last Host Frame Number																															
*11	*10	*9	*8	*7	*6	*5	*4	*19	*18	*17	*16	*15	*14	*13	*12	Reserved															

*1 *IG Mode*
*2 *Timestamp Valid*
*3 *Earth Reference Model*
*4 *IG Condition – Overframing*
*5 *IG Condition – Paging*
*6 *IG Condition – Excessive Variable Length Data*
*7 *IG Condition #4 - Reserved*
*8 *IG Condition #5 - Reserved*
*9 *IG Condition #6 - Reserved*
*10 *IG Condition #7 - Reserved*
*11 *IG Condition #8 - Reserved*
*12 *IG Condition #9 - Reserved*
*13 *IG Condition #10 - Reserved*
*14 *IG Condition #11 - Reserved*
*15 *IG Condition #12 - Reserved*
*16 *IG Condition #13 - Reserved*
*17 *IG Condition #14 - Reserved*
*18 *IG Condition #15 - Reserved*
*19 *IG Condition #16 - Reserved*

Figure 115 – Start of Frame Packet Structure

Table 48 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 48 – Start of Frame Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: FFFFh	This parameter identifies this data packet as the **Start of Frame** packet. The IG shall set this parameter to FFFFh.
Major Version Type: unsigned int8 Units: N/A Value: **4**	This parameter indicates the major version of the CIGI interface that is currently being used by the IG. The Host should use this number to determine concurrency. The IG shall set this parameter to 4.
Database Number Type: int8 Units: N/A Values: -128 Indicates database is not available -127 to -1 Identifies database being loaded 1 to 127 Identifies database that is loaded 0 Indicates IG controls database loading Default: 1	This parameter is used to indicate to the Host which database is currently in use and if that database is being loaded. The Host shall set the *Database Number* parameter of the **IG Control** packet to direct the IG to begin loading the corresponding database. The IG shall indicate that the database is being loaded by negating the value and placing it in the *Database Number* parameter of the **Start of Frame** packet. The Host shall then acknowledge this change by setting the *Database Number* parameter of the **IG Control** packet to zero (0). When the database load is complete *and* after the Host has acknowledged the database change, the IG shall set this parameter to the positive database number. The IG is now considered mission-ready. If the Host requests a database that does not exist or fails to load, the IG shall set this parameter to -128. Zero (0) shall be used to indicate that the IG controls the database loading. Refer to Section 6.1.1 for more information about the **IG Control** packet.
Minor Version Type: 4-bit field Units: N/A Value: **0**	This parameter indicates the minor version of the CIGI interface that is currently being used by the IG. The Host should use this number to determine concurrency. The IG shall set this parameter to 0.

Parameter	Description
IG Mode Type: unsigned 2-bit field Units: N/A Values: 0 Reset/Standby 1 Operate 2 Debug 3 Offline Maintenance Default: 0	This parameter indicates the current IG mode. It shall be one of the following values: **Reset/Standby** – This shall be the IG's initial state upon start-up. When set to this mode, the IG shall initialize/reinitialize the simulation. All entities that were instantiated during a previous mission shall be destroyed. All environmental and weather properties, views, view groups, components, and sensors shall revert to their default settings. Any Host-defined rates, trajectories, and collision detection segments and volumes shall be removed. All in-process mission function requests shall be purged. When the IG is in this mode the IG shall only send the Start of Frame data packet to the Host and shall ignore Host inputs except for the *IG Mode* parameter of the **IG Control** data packet. The IG shall remain in this mode until directed otherwise by the Host or the IG's user interface. **Operate** – This is the normal real-time operating mode for the IG. All packets issued by the Host shall be processed by the IG. The IG should not perform diagnostics in this mode. **Debug** – This mode is similar to the Operate mode but provides data and/or error logging and other debugging features to aid integration or troubleshooting of the Host and IG interface. Because of the overhead of these debugging features, the IG may not always operate in a hard real-time fashion. **Offline Maintenance** – In this mode, the IG shall ignore all data from the Host and shall send only **Start of Frame** packets. This mode shall be activated only from the IG. Because the **IG Control** packets from the Host are ignored by the IG, the IG shall be placed into Reset/Standby mode before the Host may initiate further mode changes.
Timestamp Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Timestamp* parameter contains a valid value. If the IG populates the *Timestamp* parameter with an accurate clock value, then the IG should set *Timestamp Valid* to Valid (1); otherwise, the IG shall set this parameter to Invalid (0). The IG should correctly populate the *Timestamp* parameter and set *Timestamp Valid* to Valid (1) in asynchronous mode.

Parameter	Description
Earth Reference Model Type: 1-bit field Units: N/A Values: 0 WGS 84 1 Host-Defined Default: 0	This parameter indicates whether the IG is using a custom (Host-defined) Earth Reference Model (ERM) or the default WGS 84 reference ellipsoid for coordinate conversion calculations. Host-defined ERMs are defined with the **Earth Reference Model Definition** packet (see Section 6.1.19). If the Host defines an ERM the IG cannot support, the IG shall set this value to WGS 84 (0). In such cases, the Host shall redefine the ERM or use the WGS 84 reference ellipsoid.
IG Frame Number Type: unsigned int32 Units: N/A	This parameter uniquely identifies an IG data frame. The IG shall increment this value by one (1) for each successive message. Note: In CIGI 3.0/3.1 this parameter was named “Frame Counter” and was incremented each frame by the IG. The Host would then return the value in the *Frame Counter* parameter of the next **IG Control** packet. As of CIGI 3.2, however, the *Host Frame Number* is independent of the *IG Frame Number* parameter in the **Start of Frame** packet.
Timestamp Type: unsigned int32 Units: 10 microseconds (μs) Datum: Arbitrary reference time	This parameter indicates the number of 10μs “ticks” since some initial reference time. The IG should use this value to correct for latencies as described in Section 4.2.1. The 10μs unit allows the simulation to run for approximately 12 hours before a timestamp rollover occurs. The Host software should contain logic to detect and correct for rollover. The IG shall populate this parameter with a valid time in asynchronous operation. The use of this parameter is optional for synchronous operation. If this parameter does not contain a valid timestamp, then the IG shall set the *Timestamp Valid* parameter to zero (0).
Last Host Frame Number Type: unsigned int32 Units: N/A	This parameter identifies the most recent Host-to-IG message received by the IG. The IG shall populate this parameter with the value of the *Host Frame Number* parameter in the last **IG Control** packet received from the Host. This parameter serves as an acknowledgement that the IG received the last message.

Parameter	Description
IG Condition - Overframing Type: 1-bit field Units: N/A Values: 0 Not present 1 Present	This parameter identifies the IG is overloaded and not able to maintain frame rate. The IG shall set this parameter to one (1) when the overframing condition is detected, otherwise, the IG shall set this parameter to zero (0).
IG Condition - Paging Type: 1-bit field Units: N/A Values: 0 Not present 1 Present	This parameter identifies the IG is asynchronously paging terrain in an asynchronous, non-blocking, continuous fashion in Operate/Debug mode and still able to process CIGI commands. The IG shall set this parameter to one (1) when the paging condition is present, otherwise, the IG shall set this parameter to zero (0). This is not to be confused with the existing “database change” handshaking communicated by way of the Database Number parameter.
IG Condition – Excessive Variable Length Data Type: 1-bit field Units: N/A Values: 0 Not present 1 Present	This parameter identifies the IG received an excessive amount of data in one or more variable length packets and is not able to maintain frame rate. The IG shall set this parameter to one (1) when the excessive variable length data condition is detected, otherwise, the IG shall set this parameter to zero (0). This condition may signify the bandwidth between the Host and IG is insufficient for the quantity of data sent or it may be accompanied by the *Overframing* condition to indicate the quantity of varying length data received has overloaded the IG. An IG setting this condition to one (1) should provide additional information about the problem via the **Image Generator Message** packet.

6.2.2 HAT/HOT Response

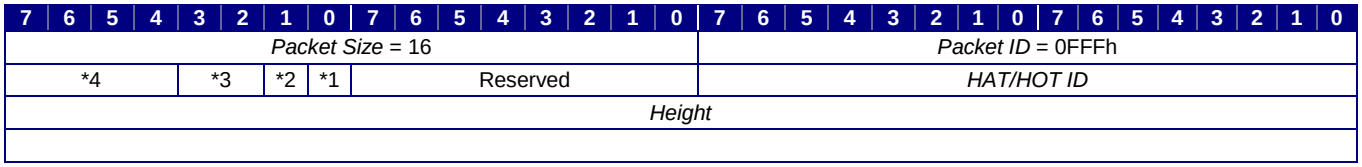
The **HAT/HOT Response** packet is sent by the IG in response to a **HAT/HOT Request** packet (Section 6.1.24) whose *Request Type* parameter was set to HAT (0) or HOT (1). This packet provides either the Height Above Terrain (HAT) or Height Of Terrain (HOT) for the test point. This packet does not contain the material code or surface normal of the terrain.

If the *Update Period* parameter of the originating **HAT/HOT Request** packet was set to a value greater than zero, then the IG shall set the *Host Frame Number LSN* parameter of each corresponding **HAT/HOT Response** packet to the least significant nybble (LSN) of the *Host Frame Number* value last received by the IG before the HAT or HOT value is calculated. The Host may correlate this LSN to an eyepoint position or may use the value to determine latency.

The IG can only return the HAT or HOT for a point that is within the bounds of the current database. If the HAT or HOT cannot be returned, the *Valid* parameter shall be set to Invalid (0).

Note that the instantaneous elevation of the water including wave displacement may be determined from a Height Of Terrain request.

The contents of the **HAT/HOT Response** packet are as follows:



*1 *Valid*
*2 *Response Type*
*3 Reserved
*4 *Host Frame Number LSN*

Figure 116 – HAT/HOT Response Packet Structure

Table 49 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 49 – HAT/HOT Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 16.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFFh	This parameter identifies this data packet as the **HAT/HOT Response** packet. The IG shall set this parameter to 0FFFh.

Parameter	Description
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Height* parameter contains a valid number. The IG shall set this value to zero (0) if the test point is beyond the database bounds.
Response Type Type: 1-bit field Units: N/A Values: 0 HAT 1 HOT	This parameter indicates whether the *Height* parameter represents Height Above Terrain or Height Of Terrain. See *Height* parameter description for details.
Host Frame Number LSN Type: unsigned 4-bit field Units: N/A	This parameter identifies the frame during which the IG calculated the HAT or HOT. The IG shall set this field to the least significant nybble of the *Host Frame Number* parameter of the last **IG Control** packet received before the HAT or HOT is calculated. This parameter shall be ignored if the *Update Period* parameter of the corresponding **HAT/HOT Request** packet was set to zero (0).
HAT/HOT ID Type: unsigned int16 Units: N/A	This parameter identifies the HAT or HOT response. The IG shall set this value to correspond to the value of the *HAT/HOT ID* parameter in the associated **HAT/HOT Request** packet.
Height Type: double float Units: meters Datum: For HAT: Terrain/Sea Surface Height For HOT: Mean Sea Level	This parameter contains the requested height. If *Request Type* is set to HAT (0), this value shall represent the Height Above Terrain. If *Request Type* is set to HOT (1), this value shall represent the Height Of Terrain. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).

6.2.3 HAT/HOT Extended Response

The **HAT/HOT Extended Response** packet is sent by the IG in response to a **HAT/HOT Request** packet (Section 6.1.24) whose *Request Type* parameter was set to Extended (2). This packet provides the Height Above Terrain (HAT) and Height Of Terrain (HOT) for the test point. This packet also contains the material code of the terrain and a surface-normal vector emanating from the terrain.

The surface-normal vector is specified as a pair of angles representing azimuth and elevation relative to a plane that is parallel to the Geodetic Reference Plane (see Section 5.4.1.2) as shown in the figure below. A vector length is not specified; the vector is assumed to be a unit vector.

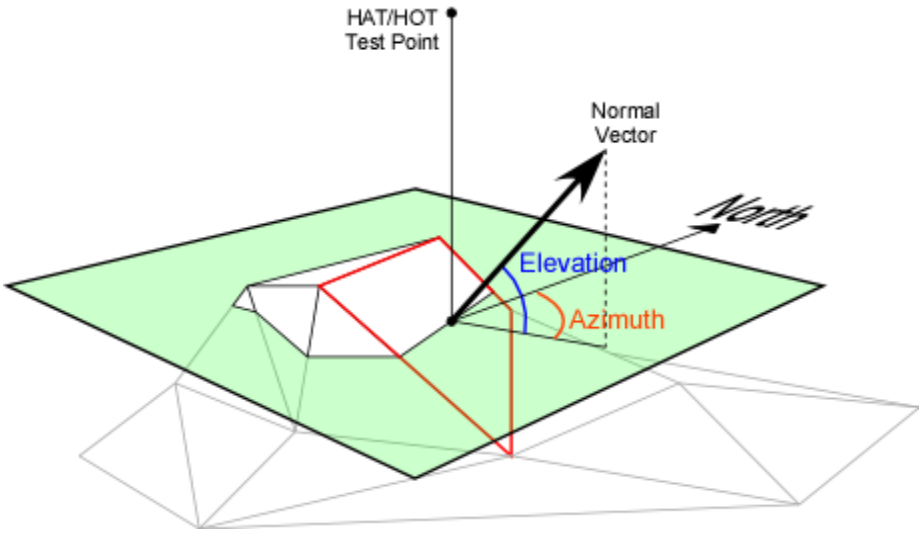


Figure 117 – HAT/HOT Extended Response Normal Vector

If the *Update Period* parameter of the originating **HAT/HOT Request** packet was set to a value greater than zero, then the IG shall set the *Host Frame Number LSN* parameter of each corresponding **HAT/HOT Response** packet to the least significant nibble of the *Host Frame Number* value last received by the IG before the HAT or HOT value is calculated. The Host may correlate this LSN to an eyepoint position or may use the value to determine latency.

The IG can only return the HAT and HOT for a point that is within the bounds of the current database. Likewise, the material code and normal vector can only be calculated within the database bounds. If these data cannot be returned, the *Valid* parameter shall be set to zero (0).

Note that the instantaneous elevation of the water including wave displacement may be determined from a Height Of Terrain request.

The contents of the **HAT/HOT Extended Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 40																*Packet ID* = 0FFEh															
*2				Reserved				*1	Reserved							*HAT/HOT ID*															
HAT																															
HOT																															
Material Code																															
Normal Vector Azimuth																															
Normal Vector Elevation																															
Reserved																															

[*1] *Valid*
[*2] *Host Frame Number LSN*

Figure 118 – HAT/HOT Extended Response Packet Structure

Table 50 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 50 – HAT/HOT Extended Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 40.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFEh	This parameter identifies this data packet as the **HAT/HOT Extended Response** packet. The IG shall set this parameter to 0FFEh.
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the remaining parameters in this packet contain valid numbers. The IG shall set this value to zero (0) if the test point is beyond the database bounds.

Parameter	Description
Host Frame Number LSN Type: unsigned 4-bit field Units: N/A	This parameter contains the least significant nybble of the *Host Frame Number* parameter of the last **IG Control** packet received before the HAT/HOT is calculated. This parameter shall be ignored if the *Update Period* parameter of the corresponding **HAT/HOT Request** packet was set to zero (0).
HAT/HOT ID Type: unsigned int16 Units: N/A	This parameter identifies the HAT/HOT response. The IG shall set this value to correspond to the value of the *HAT/HOT ID* parameter in the associated **HAT/HOT Request** packet.
HAT Type: double float Units: meters Datum: Terrain/Sea Surface Height	This parameter indicates the height of the test point above the terrain. The IG shall set this value to negative if the test point is below the terrain. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).
HOT Type: double float Units: meters Datum: Mean Sea Level	This parameter indicates the height of terrain at the test point location. This value is relative to the ellipsoid height, or Mean Sea Level. The IG shall set this value to negative if the terrain is below the datum. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).
Material Code Type: unsigned int32 Units: N/A	This parameter indicates the material code of the terrain surface at the point of intersection with the HAT/HOT test vector. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).
Normal Vector Azimuth Type: single float Units: degrees Values: -180.0 – 180.0 Datum: True North	This parameter indicates the azimuth of a vector normal to the surface intersected by the HAT/HOT test vector. The IG shall set this value to the horizontal angle from True North to the normal vector. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).

Parameter	Description
Normal Vector Elevation Type: single float Units: degrees Values: -90.0 – 90.0 Datum: Geodetic reference plane (Section 5.4.1.2)	This parameter indicates the elevation of a vector normal to the surface intersected by the HAT/HOT test vector. The IG shall set this value to the vertical angle from the geodetic reference plane to the normal vector. The Host shall ignore this parameter if the *Valid* parameter is set to zero (0).

6.2.4 Line of Sight Response

The **Line of Sight Response** packet is used in response to both the **Line of Sight Segment Request** and **Line of Sight Vector Request** packets. This packet contains the distance from a Line of Sight (LOS) segment or vector source point to the point of intersection with a polygon surface. The IG shall send this packet when the *Request Type* parameter of the originating request packet was set to Basic (0).

The IG shall send a **Line of Sight Response** packet for each intersection along the LOS segment or vector. The IG shall set the *Response Count* parameter to the total number of responses that are being returned. This allows the Host to determine when all response packets for the given request have been received.

If no intersections occurred along the segment or vector, then the IG shall send a single **Line of Sight Response** packet with the *Valid* parameter set to zero (0).

If the *Update Period* parameter of the originating **Line of Sight Segment Request** or **Line of Sight Vector Request** packet was set to a value greater than zero, then the IG shall set the *Host Frame Number LSN* parameter of each corresponding **Line of Sight Response** packet to the least significant nybble of the *Host Frame Number* value last received by the IG before the range is calculated. The Host may correlate this LSN to an eyepoint position or may use the value to determine latency.

For responses to **Line of Sight Segment Request** packets, the *Range* parameter specifies the range to the intersection point along the LOS test segment. If the destination point specified in the request is occulted, this parameter specifies the range to the surface occulting the destination. If the destination point is not occluded, this parameter simply provides the range to the destination point. Figure 120 of the Line of Sight Extended response packet illustrates this concept.

For responses to **Line of Sight Vector Request** packets, the *Range* parameter specifies the range to the point of intersection between the test vector and a surface. If no intersection occurs within the valid range specified in the request, the *Valid* parameter is set to Invalid (0). Figure 121 of the Line of Sight Extended response packet illustrates this concept.

The contents of the **Line of Sight Response** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 24</td><td colspan="16">Packet ID = 0FFD</td></tr>
<tr><td colspan="16">LOS ID</td><td colspan="16">Entity ID</td></tr>
<tr><td colspan="8">*5</td><td>*4</td><td>*3</td><td>*2</td><td>*1</td><td colspan="4">Response Count</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="32">Reserved</td></tr>
<tr><td colspan="32">Range</td></tr>
<tr><td colspan="32"></td></tr>
</table>

*1 *Valid*
*2 *Entity ID Valid*
*3 *Visible*
*4 Reserved
*5 *Host Frame Number LSN*

Figure 119 – Line of Sight Response Packet Structure

Table 51 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 51 – Line of Sight Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFDh	This parameter identifies this data packet as the **Line of Sight Response** packet. The IG shall set this parameter to 0FFDh.
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS response. The IG shall set this field to the value of the *LOS ID* parameter in the associated **Line of Sight Segment Request** packet (Section 6.1.25) or **Line of Sight Vector Request** packet (Section 6.1.26).
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity with which an LOS test vector or segment intersected. The IG shall set this value to the Entity ID of the entity that was intersected during this test. Note: There may be multiple responses with different entity IDs. The Host shall ignore this parameter if *Entity ID Valid* is set to Invalid (0).
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Range* parameter is valid. For a response to a LOS vector request packet, the IG shall set this to Invalid (0) if no intersection occurred, or if an intersection occurred before the minimum range or beyond the specified maximum range. The IG shall set this parameter to Valid (1) if one or more intersections occurred. For a response to a LOS segment request, the IG shall set this to Valid (1) unless the specified source or destination entity does not exist. For both types of request, the IG shall set this to Invalid (0) if a nonexistent entity was specified in the originating request.

Parameter	Description
Entity ID Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the LOS test vector or segment intersects with an entity. The IG shall set this parameter to Invalid (0) if the intersected polygon was not part of an entity. The IG shall set this parameter to Valid (1) if the intersected polygon was part of an entity.
Visible Type: 1-bit field Units: N/A Values: 0 Occluded (not visible) 1 Visible	This parameter indicates whether the destination point specified in a **Line of Sight Segment** request packet is visible from the source point. The IG shall set this parameter to Occluded (0) if the segment intersected one or more polygons before reaching the destination point. If the LOS segment destination point is within the body of a target entity model, then the IG shall set this parameter to Occluded (0) and the *Entity ID* parameter to the ID of that entity The IG shall set this parameter to Visible (1) if the segment did not intersect with any polygons before reaching the destination point. The Host shall ignore this parameter if the packet is in response to a **Line of Sight Vector Request** packet.
Host Frame Number LSN Type: unsigned 4-bit field Units: N/A	This parameter contains the least significant nybble of the *Host Frame Number* parameter of the last **IG Control** packet received before the LOS data are calculated. The Host shall ignore this parameter if the *Update Period* parameter of the corresponding **Line of Sight Segment Request** or **Line of Sight Vector Request** packet was set to zero (0).
Response Count Type: unsigned int8 Units: N/A	This parameter indicates the total number of **Line of Sight Response** packets the IG shall return for the corresponding request. If *Visible* is set to Occluded (0), The IG shall set the *Response Count* to the number of reportable intersections. Note: An intersection will not be reported if a material has been excluded from intersection testing via the *Material Mask* value of the **Line of Sight Segment Request** or **Line of Sight Vector Request** packets. If *Visible* is set to Visible (1), the IG shall set the *Response Count* to 1.

Parameter	Description
Range Type: double float Units: meters Datum: LOS vector or segment source point	This parameter indicates the distance along the LOS test segment or vector from the source point to the point of intersection with a polygon surface. The IG shall set this value to the distance between the test's segment or vector source point and the point with which the LOS test vector or segment intersected. If there was no intersection, and the test was a segment test, then the IG shall set this range to the distance between the source and destination points.

6.2.5 Line of Sight Extended Response

The **Line of Sight Extended Response** packet is used in response to both **Line of Sight Segment Request** and **Line of Sight Vector Request** packets. This packet contains positional data describing the Line of Sight (LOS) intersection point (see Section 6.2.4 for details on these data). In addition, it contains the material code and surface-normal vector emanating from the polygon at the point of intersection (see Page 275, Figure 117). The IG shall send this packet when the *Request Type* parameter of the originating request packet was set to Extended (1).

The IG shall send a **Line of Sight Extended Response** packet for each intersection along the LOS segment or vector. The IG shall set the *Response Count* parameter to the total number of responses that are being returned. This allows the Host to determine when all response packets for the given request have been received.

For responses to **Line of Sight Segment Request** packets, the *Range*, *Altitude*, *Latitude*, and *Longitude* parameters specify the range to and position of the intersection point along the LOS test segment. If the destination point specified in the request is occulted, these parameters specify the range to and position of a point on the surface occulting the destination. If the destination point is not occluded, these parameters simply provide the range to and position of the destination point. Figure 120 illustrates two LOS test segments and the data returned with the responses:

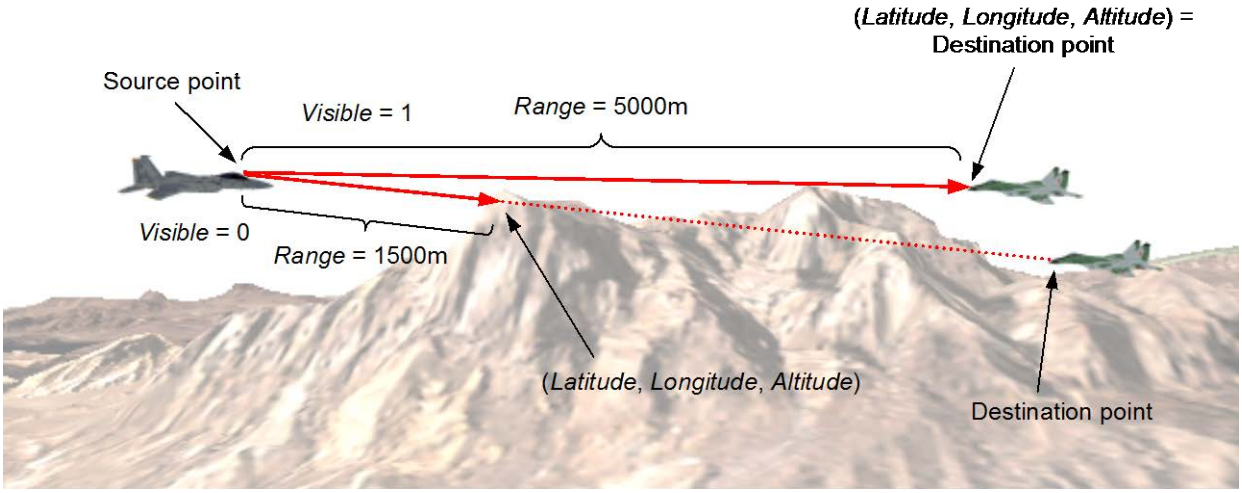


Figure 120 – Responses to Line of Sight Segment Requests

For responses to **Line of Sight Vector Request** packets, the *Range*, *Altitude*, *Latitude*, and *Longitude* parameters specify the range to and position of the point of intersection between the test vector and a surface. If no intersection occurs within the valid range specified in the request, the *Valid* parameter is set to Invalid (0). Figure 121 illustrates two LOS test vectors and the data returned with the responses:

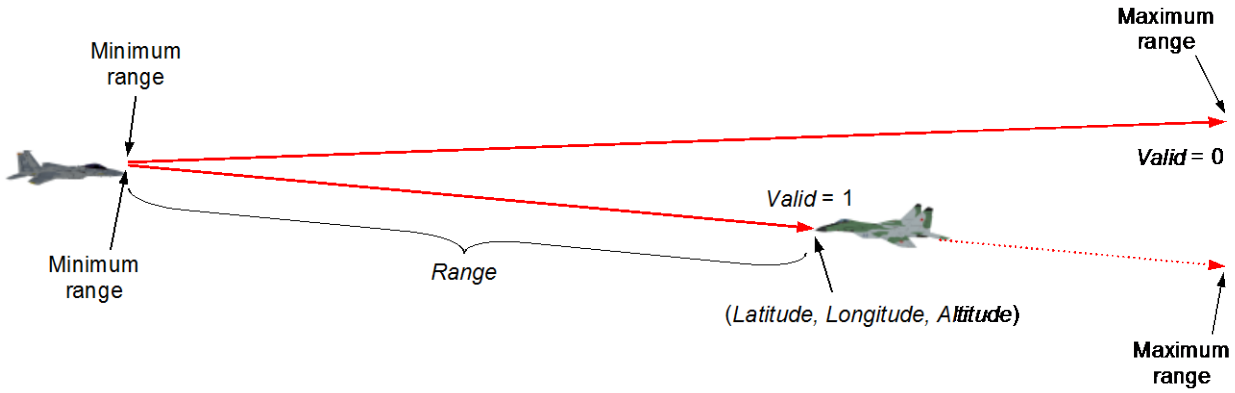


Figure 121 – Responses to Line of Sight Vector Requests

If the *Update Period* parameter of the originating **Line of Sight Segment Request** or **Line of Sight Vector Request** packet was set to a value greater than zero, then the IG shall set the *Host Frame Number LSN* parameter of each corresponding **Line of Sight Response** packet to the least significant nybble of the *Host Frame Number* value last received by the IG before the range is calculated. The Host may correlate this LSN to an eyepoint position or may use the value to determine latency.

The contents of the **Line of Sight Extended Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 64																*Packet ID* = 0FFCh															
LOS ID																*Entity ID*															
*5				*4	*3	*2	*1	*Response Count*								Reserved															
Reserved																															
Range																															
Latitude/X Offset																															
Longitude/Y Offset																															
Altitude/Z Offset																															
Red								*Green*								*Blue*								*Alpha*							
Material Code																															
Normal Vector Azimuth																															
Normal Vector Elevation																															

*1 *Valid*
*2 *Entity ID Valid*
*3 *Range Valid*
*4 *Visible*
*5 *Host Frame Number LSN*

Figure 122 – Line of Sight Extended Response Packet Structure

Table 52 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 52 – Line of Sight Extended Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 64	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 64.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFCh	This parameter identifies this data packet as the **Line of Sight Extended Response** packet. The IG shall set this parameter to 0FFCh.
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS response. The IG shall set this field to the value of the *LOS ID* parameter in the associated **Line of Sight Segment Request** packet (Section 6.1.25) or **Line of Sight Vector Request** packet (Section 6.1.26).
Entity ID Type: unsigned int16 Units: N/A	The value of this parameter specifies the entity with which an LOS test vector or segment intersected. The IG shall set this value to the Entity ID of the entity that was intersected during this test. Note: There may be multiple responses with different entity IDs. The Host shall ignore this parameter if *Entity ID Valid* is set to Invalid (0).
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Range* parameter is valid. For a response to a LOS vector request packet, the IG shall set this to Invalid (0) if no intersection occurred, or if an intersection occurred before the minimum range or beyond the specified maximum range. The IG shall set this parameter to Valid (1) if one or more intersections occurred. For a response to a LOS segment request, the IG shall set this to Valid (1) unless the specified source or destination entity does not exist. For both types of request, the IG shall set this to Invalid (0) if a nonexistent entity was specified in the originating request.

Parameter	Description
Entity ID Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the LOS test vector or segment intersects with an entity. The IG shall set this parameter to Invalid (0) if the intersected polygon was not part of an entity. The IG shall set this parameter to Valid (1) if the intersected polygon was part of an entity.
Range Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Range* parameter is valid. The range shall be invalid if an intersection occurs before the minimum range or beyond the maximum range specified in an LOS vector request. The range shall also be invalid if this packet is in response to an LOS segment request. If *Valid* is set to Invalid (0), this parameter shall also be set to Invalid (0).
Visible Type: 1-bit field Units: N/A Values: 0 Occluded (not visible) 1 Visible	This parameter indicates whether the destination point specified in a **Line of Sight Segment** request packet is visible from the source point. The IG shall set this parameter to Occluded (0) if the segment intersected one or more polygons before reaching the destination point. If the LOS segment destination point is within the body of a target entity model, then the IG shall set this parameter to Occluded (0) and the *Entity ID* parameter to the ID of that entity The IG shall set this parameter to Visible (1) if the segment did not intersect with any polygons before reaching the destination point. The Host shall ignore this parameter if the packet is in response to a **Line of Sight Vector Request** packet.
Host Frame Number LSN Type: unsigned 4-bit field Units: N/A	This parameter contains the least significant nybble of the *Host Frame Number* parameter of the last **IG Control** packet received before the LOS data are calculated. The Host shall ignore this parameter if the *Update Period* parameter of the corresponding **Line of Sight Segment Request** or **Line of Sight Vector Request** packet was set to zero (0).

Parameter	Description
Response Count Type: unsigned int8 Units: N/A	This parameter indicates the total number of **Line of Sight Response** packets the IG shall return for the corresponding request. If *Visible* is set to Occluded (0), The IG shall set the *Response Count to* the number of reportable intersections. Note: An intersection shall not be reported if a material has been excluded from intersection testing via the *Material Mask* value of the **Line of Sight Segment Request** or **Line of Sight Vector Request** packets. If *Visible* is set to Visible (1), the IG shall set the *Response Count* to 1.
Range Type: double float Units: meters Datum: LOS vector or segment source point	This parameter indicates the distance along the LOS test segment or vector from the source point to the point of intersection with a polygon surface. The IG shall set this value to the distance between the test's segment or vector source point and the point with which the LOS test vector or segment intersected. If there was no intersection, and the test was a segment test, then the IG shall set this range to the distance between the source and destination points.
Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90 – 90 Datum: Equator	If the *Entity ID Valid* parameter is set to Invalid (0) or if *Response Coordinate System* in the requesting packet was set to Geodetic (0), the IG shall set the value of this parameter to the geodetic latitude of the point of intersection along the LOS test segment or vector. If this packet is in response to an LOS segment request and *Visible* is set to Occluded (0), this point is on the occulting surface. If this packet is in response to an LOS segment request and *Visible* is set to Visible (1), this point is simply the destination point.
X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If the *Entity ID Valid* parameter is set to Valid (1) and *Response Coordinate System* in the requesting packet was set to Entity (1), the IG shall set the value of this parameter to the offset of the point of intersection of the LOS test segment or vector along the intersected entity's **X** axis.

Parameter	Description
Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180 – 180 Datum: Prime Meridian	If the *Entity ID Valid* parameter is set to Invalid (0) or if *Response Coordinate System* in the requesting packet was set to Geodetic (0), the IG shall set the value of this parameter to the geodetic longitude of the point of intersection along the LOS test segment or vector. If this packet is in response to an LOS segment request and *Visible* is set to Occluded (0), this point is on the occulting surface. If this packet is in response to an LOS segment request and *Visible* is set to Visible (1), this point is simply the destination point.
Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If the *Entity ID Valid* parameter is set to Valid (1) and *Response Coordinate System* in the requesting packet was set to Entity (1), the IG shall set the value of this parameter to the offset of the point of intersection of the LOS test segment or vector along the intersected entity's **Y** axis.
Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If the *Entity ID Valid* parameter is set to Invalid (0) or if *Response Coordinate System* in the requesting packet was set to Geodetic (0), the IG shall set the value of this parameter to the geodetic altitude of the point of intersection along the LOS test segment or vector. If this packet is in response to a LOS segment request and *Visible* is set to Occluded (0), this point is on the occulting surface. If this packet is in response to a LOS segment request and *Visible* is set to Visible (1), this point is simply the destination point.
Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity reference point	If the *Entity ID Valid* parameter is set to Valid (1) and *Response Coordinate System* in the requesting packet was set to Entity (1), the IG shall set the value of this parameter to the offset of the point of intersection of the LOS test segment or vector along the intersected entity's **Z** axis.
Red Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the red color component of the surface at the point of intersection.

Parameter	Description
Green Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the green color component of the surface at the point of intersection.
Blue Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the blue color component of the surface at the point of intersection.
Alpha Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the alpha component of the surface at the point of intersection.
Material Code Type: unsigned int32 Units: N/A	The IG shall set the value of this parameter to the material code of the surface intersected by the LOS test segment or vector.
Normal Vector Azimuth Type: single float Units: degrees Values: -180.0 – 180.0 Datum: True North	This parameter indicates the azimuth of a vector normal to the surface intersected by the LOS test segment or vector. The IG shall set this value to the horizontal angle from True North to the normal vector. The Host shall only consider this parameter valid if the *Valid* parameter is set to one (1).
Normal Vector Elevation Type: single float Units: degrees Values: -90.0 – 90.0 Datum: Geodetic reference plane	This parameter indicates the elevation of a vector normal to the surface intersected by the LOS test segment or vector. The IG shall set this value to the vertical angle from the geodetic reference plane to the normal vector. The Host shall only consider this parameter is valid only if the *Valid* parameter is set to one (1).

6.2.6 Sensor Response

The **Sensor Response** packet is used to report the gate size and position on a sensor display to the Host.

The sensor gate size and position are defined with respect to the 2D view coordinate system. The +**X** axis is to the right of the screen and the +**Y** axis is up. The origin is at the intersection of the viewing vector with the view plane. The gate position is measured in degrees along each axis from the origin to the center of the gate as shown below:

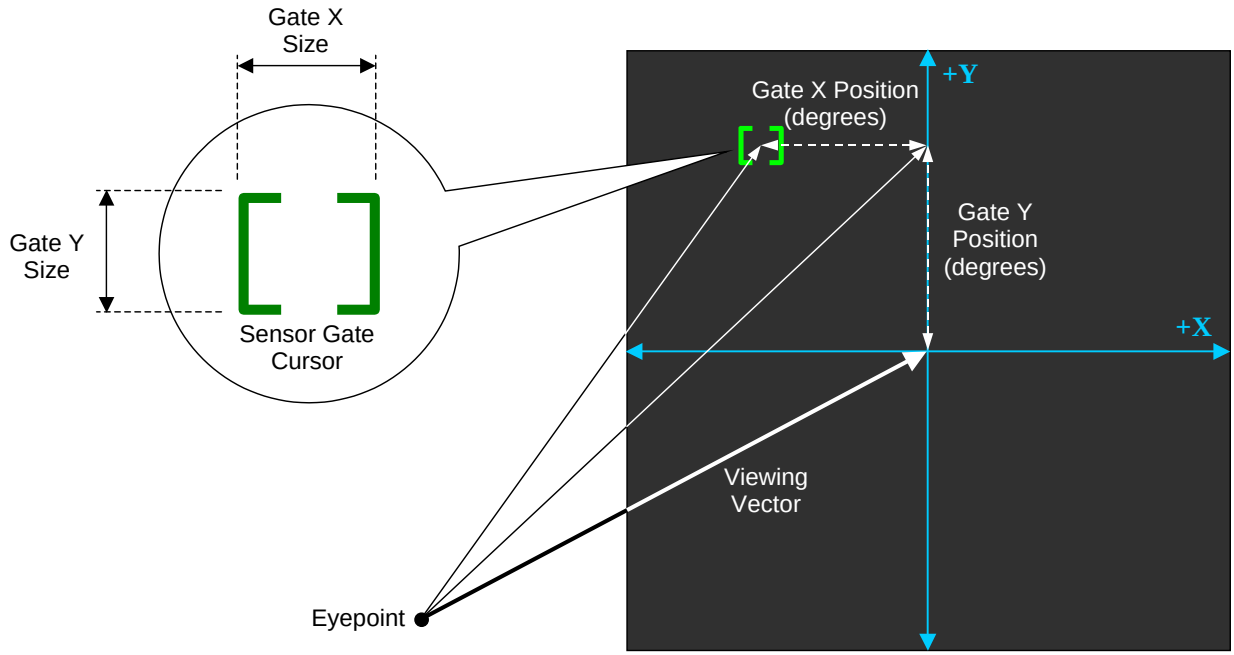


Figure 123 – Sensor Gate Size and Position

The *Gate X Position* and *Gate Y Position* angles correspond to the horizontal and vertical angles formed between the sensor's viewing vector and a vector from the sensor eyepoint to the track point. Scaling of the sensor view may be performed with a **View Definition** packet (Section 6.1.21).

The *Host Frame Number* parameter contains the value of the *Host Frame Number* parameter of the **IG Control** packet (see Section 6.1.1) last received by the IG before the gate and line-of-sight intersection data are calculated. The Host may correlate this value to an eyepoint position or may use the value to determine sensor sampling rate latency.

Either this packet or the **Sensor Extended Response** packet (Section 6.2.7) shall be sent to the Host during each frame that the specified sensor is active.

The contents of the **Sensor Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = 0FFBh															
Sensor ID								Reserved						*1		*View ID*															
Gate X Size																*Gate Y Size*															
Gate X Position																															
Gate Y Position																															
Host Frame Number																															

*1 *Sensor Status*

Figure 124 – Sensor Response Packet Structure

Table 53 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 53 – Sensor Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFBh	This parameter identifies this data packet as the **Sensor Response** packet. The IG shall set this parameter to 0FFBh .
Sensor ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the sensor to which the data in this packet apply.
Sensor Status Type: unsigned 2-bit field Units: N/A Values: 0 Searching for target 1 Tracking target 2 Impending breaklock 3 Breaklock	The IG shall set the value of this parameter to the current tracking state of the sensor.

Parameter	Description
View ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the view that represents the sensor display.
Gate X Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	The IG shall set the value of this parameter to the gate symbol size along the view's **X** axis. Note: This size is specified in either pixels or raster lines depending upon the orientation of the display.
Gate Y Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	The IG shall set the value of this parameter to the gate symbol size along the view's **Y** axis. Note: This size is specified in either pixels or raster lines depending upon the orientation of the display.
Gate X Position Type: single float Units: degrees Datum: Viewing vector	This parameter specifies the gate symbol's position along the view's **X** axis. The IG shall set the value of this parameter to the horizontal angle formed at the sensor eyepoint between the sensor's viewing vector and the center of the track point.
Gate Y Position Type: single float Units: degrees Datum: Viewing vector	This parameter specifies the gate symbol's position along the view's **Y** axis. The IG shall set the value of this parameter to the vertical angle formed at the sensor eyepoint between the sensor's viewing vector and the center of the track point.
Host Frame Number Type: unsigned int32 Units: N/A	The IG shall set the value of this parameter to the *Host Frame Number* parameter of the last **IG Control** packet received before the IG calculates the gate and line-of-sight intersection data.

6.2.7 Sensor Extended Response

The **Sensor Extended Response** packet, like the **Sensor Response** packet (Section 6.2.6), is used to report the gate size and position on a sensor display to the Host. This packet also contains the geodetic position of the sensor track point and the entity ID of the target.

Either this packet or the **Sensor Response** packet shall be sent to the Host during each frame that the specified sensor is active.

The contents of the **Sensor Extended Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 56																*Packet ID* = 0FFAh															
View ID																*Entity ID*															
Sensor ID								Reserved						*2	*1	Reserved															
Gate X Size																*Gate Y Size*															
Gate X Position																															
Gate Y Position																															
Host Frame Number																															
Reserved																															
Track Point Latitude																															
Track Point Longitude																															
Track Point Altitude																															

*1 *Sensor Status*
*2 *Entity ID Valid*

Figure 125 – Sensor Extended Response Packet Structure

Table 54 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 54 – Sensor Extended Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 56	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 56.
Packet ID Type: unsigned int16 Units: N/A Value: 0FFAh	This parameter identifies this data packet as the **Sensor Extended Response** packet. The IG shall set this parameter to 0FFAh.

Parameter	Description
View ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the view that represents the sensor display.
Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity ID of the target. The Host shall ignore this parameter if *Entity ID Valid* is set to Invalid (0).
Sensor ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the sensor to which the data in this packet apply.
Sensor Status Type: unsigned 2-bit field Units: N/A Values: 0 Searching for target 1 Tracking target 2 Impending breaklock 3 Breaklock	The IG shall set the value of this parameter to the current tracking state of the sensor.
Entity ID Valid Type: 1-bit field Units: N/A Values: 0 Invalid (non-entity target) 1 Valid (entity target)	This parameter indicates whether the target is an entity or a non-entity object. The IG shall set this parameter to Invalid (0) if the target point being tracked is not part of an entity. The IG shall set this parameter to Valid (1) if the target point being tracked is part of an entity. If the *Sensor Status* is set to Searching for target (0) the IG shall set this value to Invalid (0).
Gate X Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	The IG shall set the value of this parameter to the gate symbol size along the view's **X** axis. Note: This size is specified in either pixels or raster lines depending upon the orientation of the display.
Gate Y Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	The IG shall set the value of this parameter to the gate symbol size along the view's **Y** axis. Note: This size is specified in either pixels or raster lines depending upon the orientation of the display.

Parameter	Description
Gate X Position Type: single float Units: degrees Datum: Viewing vector	This parameter specifies the gate symbol's position along the view's **X** axis. The IG shall set the value of this parameter to the horizontal angle formed at the sensor eyepoint between the sensor's viewing vector and the center of the track point.
Gate Y Position Type: single float Units: degrees Datum: Viewing vector	This parameter specifies the gate symbol's position along the view's **Y** axis. The IG shall set the value of this parameter to the vertical angle formed at the sensor eyepoint between the sensor's viewing vector and the center of the track point.
Host Frame Number Type: unsigned int32 Units: N/A	The IG shall set the value of this parameter to the *Host Frame Number* parameter of the last **IG Control** packet received before the IG calculates the gate and line-of-sight intersection data.
Track Point Latitude Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	The IG shall set the value of this parameter to the geodetic latitude of the point being tracked by the sensor. The Host shall ignore this parameter if the *Sensor Status* parameter is set to either Searching for target (0) or Breaklock (3).
Track Point Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	The IG shall set the value of this parameter to the geodetic longitude of the point being tracked by the sensor. The Host shall ignore this parameter if the *Sensor Status* parameter is set to either Searching for target (0) or Breaklock (3).
Track Point Altitude Type: double float Units: meters Datum: Mean Sea Level	The IG shall set the value of this parameter to the geodetic altitude of the point being tracked by the sensor. The Host shall ignore this parameter if the *Sensor Status* parameter is set to either Searching for target (0) or Breaklock (3).

6.2.8 Position Response

The **Position Response** packet is sent by the IG in response to a **Position Request** packet (Section 6.1.27). This packet describes the position and orientation of an entity, articulated part, view, view group, or motion tracker.

The contents of the **Position Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 48																*Packet ID* = 0FF9h															
Articulated Part ID								Reserved				*2				*1								*Object ID*							
Latitude/X Offset																															
Longitude/Y Offset																															
Altitude/Z Offset																															
Roll																															
Pitch																															
Yaw																															
Reserved																															

*1 *Object Class*
*2 *Coordinate System*

Figure 126 – Position Response Packet Structure

Table 55 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 55 – Position Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 48.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF9h	This parameter identifies this data packet as the **Position Response** packet. The IG shall set this parameter to 0FF9h.

Parameter	Description
Articulated Part ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the articulated part whose position is being reported. The entity to which the part belongs is specified by the *Object ID* parameter. The Host shall ignore this parameter if the *Object Class* is not set to Articulated Part (1).
Object Class Type: unsigned 3-bit field Units: N/A Values: 0 Entity 1 Articulated Part 2 View 3 View Group 4 Motion Tracker	The IG shall set the value of this parameter to the type of object whose position is being reported.
Coordinate System Type: unsigned 2-bit field Units: N/A Values: 0 Geodetic 1 Parent Entity 2 Submodel	The value of this parameter indicates the coordinate system in which the position and orientation are specified. **Geodetic** – Position shall be specified as a geodetic latitude, longitude, and altitude. Orientation shall be given with respect to the reference plane shown in Figure 18, page 46. **Parent Entity** – Position and orientation shall be with respect to the entity to which the specified entity or view is attached. This value shall be invalid for top-level entities. **Submodel** – Position and orientation shall be specified with respect to the articulated part's reference coordinate system as described in Section 5.4.3. This value shall be valid only when *Object Class* is set to Articulated Part (1). Note: If *Object Class* is set to Motion Tracker (4), the coordinate system is defined by the tracking device and this parameter shall be ignored.
Object ID Type: unsigned int16 Units: N/A Values: If *Object Class* = 0: 0 – 65,535 If *Object Class* = 1: 0 – 65,535 If *Object Class* = 2: 0 – 65,535 If *Object Class* = 3: 1 – 255 If *Object Class* = 4: 0 – 255	The IG shall set the value of this parameter to the entity, view, view group, or motion tracking device whose position is being reported. If *Object Class* is set to Articulated Part (1), the IG shall set the value of this parameter to the entity whose part is identified by the *Articulated Part ID* parameter.

Parameter	Description
Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	If *Coordinate System* is set to Geodetic (0), the IG shall set the value of this parameter to the geodetic latitude of the entity, view, or view group.
X Offset (All other coordinate systems) Type: double float Units: meters Datum: If *Coordinate System* = 1: Parent entity's reference point If *Coordinate System* = 2: Submodel's reference point If *Object Class* = 4: Tracker boresight	If *Coordinate System* is set to Parent Entity (1), the IG shall set the value of this parameter to the X offset from the parent entity's origin to the child entity, articulated part, view, or view group. If *Coordinate System* is set to Submodel (2), the IG shall set the value of this parameter to the X offset from the articulated part submodel's reference point (see Section 5.4.3) If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the X position reported by the tracking device.
Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	If *Coordinate System* is set to Geodetic (0), the IG shall set the value of this parameter to the geodetic longitude of the entity, articulated part, view, or view group.
Y Offset (All other coordinate systems) Type: double float Units: meters Datum: If *Coordinate System* = 1: Parent entity's reference point If *Coordinate System* = 2: Submodel's reference point If *Object Class* = 4: Tracker boresight	If *Coordinate System* is set to Parent Entity (1), the IG shall set the value of this parameter to the Y offset from the parent entity's origin to the child entity, articulated part, view, or view group. If *Coordinate System* is set to Submodel (2), the IG shall set the value of this parameter to the Y offset from the articulated part submodel's reference point (see Section 5.4.3) If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the Y position reported by the tracking device.

Parameter	Description
Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	If *Coordinate System* is set to Geodetic (0), the IG shall set the value of this parameter to the geodetic altitude of the entity, articulated part, view, or view group.
Z Offset (All other coordinate systems) Type: double float Units: meters Datum: If *Coordinate System* = 1: Parent entity's reference point If *Coordinate System* = 2: Submodel's reference point If *Object Class* = 4: Tracker boresight	If *Coordinate System* is set to Parent Entity (1), the IG shall set the value of this parameter to the Z offset from the parent entity's origin to the child entity, articulated part, view, or view group. If *Coordinate System* is set to Submodel (2), the IG shall set the value of this parameter to the Z offset from the articulated part submodel's reference point (see Section 5.4.3) If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the Z position reported by the tracking device.
Roll Type: single float Units: degrees Values: -180.0 – 180.0 Datum: If *Coordinate System* = 0: Geodetic reference plane If *Coordinate System* = 1: Parent entity's local coordinate system If *Coordinate System* = 2: Submodel's reference plane If *Object Class* = 4: Tracker boresight	The IG shall set the value of this parameter to the roll angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the roll angle reported by the tracking device.

Parameter	Description
Pitch Type: single float Units: degrees Values: -90.0 – 90.0 Datum: If *Coordinate System* = 0: Geodetic reference coordinate system If *Coordinate System* = 1: Parent entity's local coordinate system If *Coordinate System* = 2: Submodel's reference coordinate system If *Object Class* = 4: Tracker boresight	The IG shall set the value of this parameter to the pitch angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the pitch angle reported by the tracking device.
Yaw Type: single float Units: degrees Values: 0.0 – 360.0 Datum: If *Coordinate System* = 0: Geodetic reference plane If *Coordinate System* = 1: Parent entity's local coordinate system If *Coordinate System* = 2: Submodel's reference plane If *Object Class* = 4: Tracker boresight	The IG shall set the value of this parameter to the yaw angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), the IG shall set the value of this parameter to the yaw angle reported by the tracking device.

6.2.9 Weather Conditions Response

The **Weather Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 6.1.28) whose *Request Type* parameter specifies Weather Conditions. The packet describes atmosphere properties at the requested geodetic position.

Figure 127 illustrates a hypothetical scenario wherein the Host requests the weather conditions at two points, A and B, and the IG returns a response for each point:

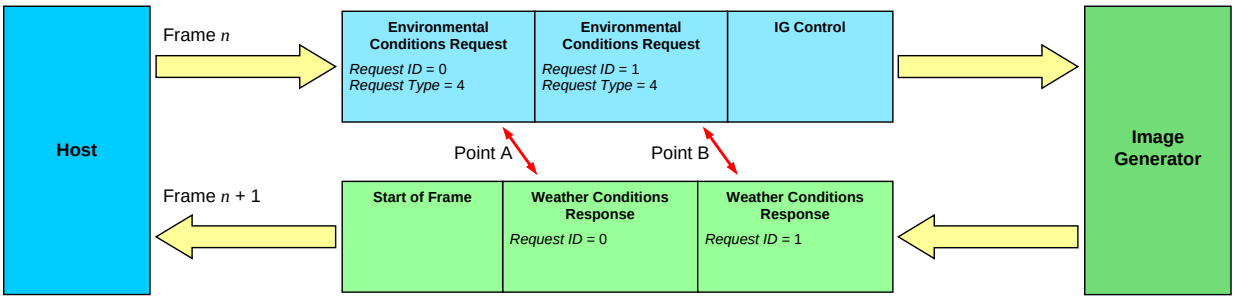


Figure 127 – Data Exchange for Weather Conditions Request

In the above example, the *Request ID* parameter of each **Environmental Conditions Request** packet is unique and is equal to the *Request ID* parameter in the corresponding **Weather Conditions Response** packet.

The contents of the **Weather Conditions Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 32																*Packet ID* = 0FF8h															
Request ID								*Humidity*								Reserved															
Air Temperature																															
Visibility Range																															
Horizontal Wind Speed																															
Vertical Wind Speed																															
Wind Direction																															
Barometric Pressure																															

Figure 128 – Weather Conditions Response Packet Structure

Table 56 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 56 – Weather Conditions Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 32.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 0FF8h	This parameter identifies this data packet as the **Weather Conditions Response** packet. The IG shall set this parameter to 0FF8h.
Request ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the environmental conditions request to which this response packet corresponds.
Humidity Type: unsigned int8 Units: percent Values: 0 – 100	The IG shall set the value of this parameter to the humidity at the requested location.
Air Temperature Type: single float Units: degrees Celsius (°C)	The IG shall set the value of this parameter to the air temperature at the requested location.
Visibility Range Type: single float Units: meters Values: ≥ 0	The IG shall set the value of this parameter to the visibility range at the requested location.
Horizontal Wind Speed Type: single float Units: m/s Values: ≥ 0 Default: 0	The IG shall set the value of this parameter to the local wind speed parallel to the ellipsoid-tangential reference plane.
Vertical Wind Speed Type: single float Units: m/s Default: 0	The IG shall set the value of this parameter to the local vertical wind speed. The IG shall set this parameter such that a positive value indicates an updraft, while a negative value indicates a downdraft.

Parameter	Description
Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	The IG shall set the value of this parameter to the local wind direction. This shall be the direction from which the wind is blowing.
Barometric Pressure Type: single float Units: millibars (mb) or hectopascals (hPa) Values: ≥ 0	The IG shall set the value of this parameter to the atmospheric pressure at the requested location.

6.2.10 Aerosol Concentration Response

The **Aerosol Concentration Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 6.1.28) whose *Request Type* parameter specifies Aerosol Concentrations. The packet describes the concentration of airborne particles associated with a specific weather layer.

The aerosol type is determined by the weather layer ID. If two or more global or regional weather layers overlap and have the same layer ID, the concentration of that aerosol is the average of the concentrations due to each layer.

Figure 129 illustrates a hypothetical scenario wherein the Host requests the aerosol concentrations at two points, A and B, and the IG returns one or more responses for each point:

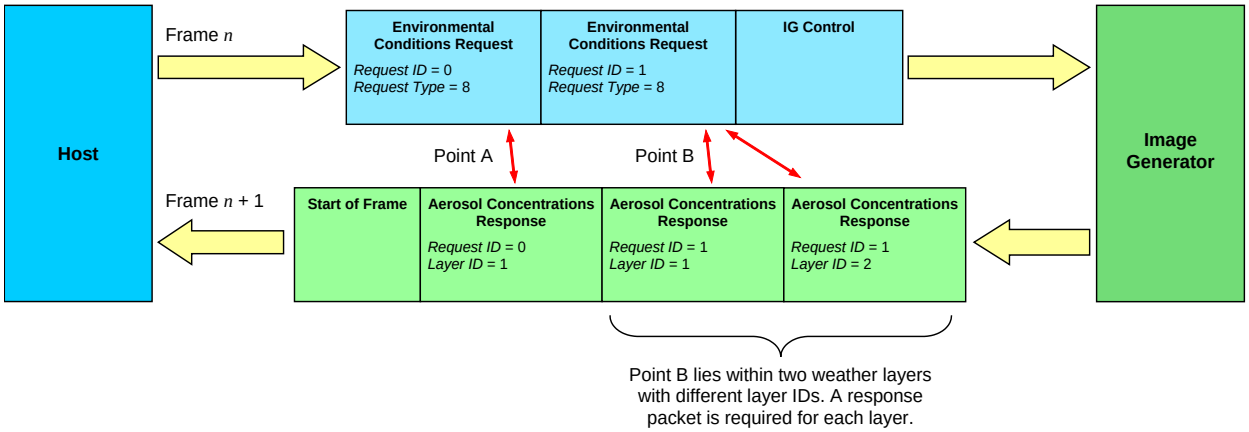


Figure 129 – Data Exchange for Aerosol Concentrations Request

In the above example, the *Request ID* parameter of each **Environmental Conditions Request** packet is unique and is equal to the *Request ID* parameter in the corresponding **Aerosol Concentration Response** packet or packets. Because Point B is located within two distinct weather layers, two separate response packets are required.

The contents of the **Aerosol Concentration Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 16																*Packet ID* = 0FF7h															
Request ID								*Layer ID*								Reserved															
Aerosol Concentration																															
Reserved																															

Figure 130 – Aerosol Concentration Response Packet Structure

Table 57 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 57 – Aerosol Concentration Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 16.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF7h	This parameter identifies this data packet as the **Aerosol Concentration Response** packet. The IG shall set this parameter to 0FF7h.
Request ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the environmental conditions request to which this response packet corresponds.
Layer ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the weather layer whose aerosol concentration is being described. Thus, this parameter indicates the aerosol type to which this packet corresponds.
Aerosol Concentration Type: single float Units: g/m^3 Values: ≥ 0	The IG shall set the value of this parameter to the concentration of airborne particles. The type of particle is identified by the *Layer ID* parameter.

6.2.11 Maritime Surface Conditions Response

The **Maritime Surface Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 6.1.28) whose *Request Type* parameter specifies Maritime Surface Conditions. The packet describes the sea surface state at the requested geodetic latitude and longitude.

Figure 131 illustrates a hypothetical scenario wherein the Host requests the maritime surface conditions at two points, A and B, and the IG returns a response for each point:

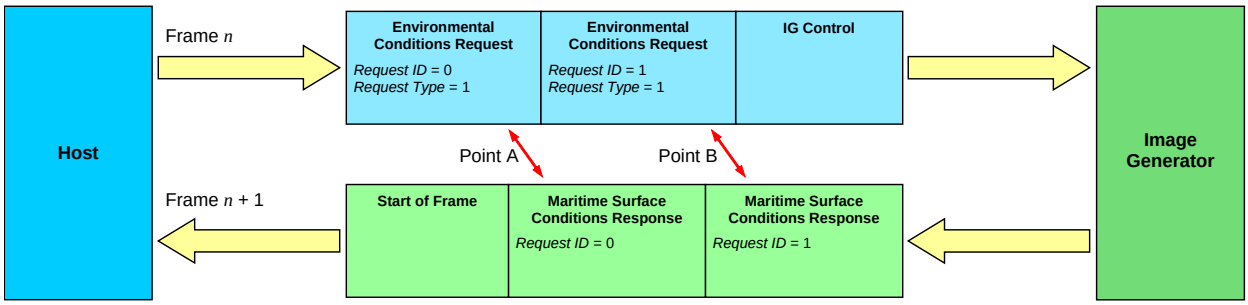


Figure 131 – Data Exchange for Maritime Surface Conditions Request

In the above example, the *Request ID* parameter of each **Environmental Conditions Request** packet is unique and is equal to the *Request ID* parameter in the corresponding **Maritime Surface Conditions Response** packet.

The contents of the **Maritime Surface Conditions Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = 0FF6h															
Request ID								Reserved																							
Sea Surface Height																															
Surface Water Temperature																															
Surface Clarity																															
Reserved																															

Figure 132 – Maritime Surface Conditions Response Packet Structure

Table 58 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 58 – Maritime Surface Conditions Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 0FF6h	This parameter identifies this data packet as the **Maritime Surface Conditions Response** packet. The IG shall set this parameter to 0FF6h.
Request ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the environmental conditions request to which this response packet corresponds.
Sea Surface Height Type: single float Units: meters Datum: Mean Sea Level	The IG shall set the value of this parameter to the height of the sea surface at equilibrium (i.e.,, without waves). Note that the instantaneous elevation of the water including wave displacement may be determined from a Height Of Terrain request.
Surface Water Temperature Type: single float Units: degrees Celsius (°C)	The IG shall set the value of this parameter to the water temperature on the sea surface at the requested location.
Surface Clarity Type: single float Units: percent Values: 0 – 100	The IG shall set the value of this parameter to the water's clarity on the sea surface at the requested location. The IG shall set this parameter such that a value of 100% indicates pristine water, while a value of 0% indicates extremely turbid water.

6.2.12 Terrestrial Surface Conditions Response

The **Terrestrial Surface Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 6.1.28) whose *Request Type* parameter specifies Terrestrial Surface Conditions. The packet describes the terrain surface conditions at the requested geodetic latitude and longitude.

Figure 133 illustrates a hypothetical scenario wherein the Host requests the terrain surface conditions at two points, A and B, and the IG returns one or more responses for each point:

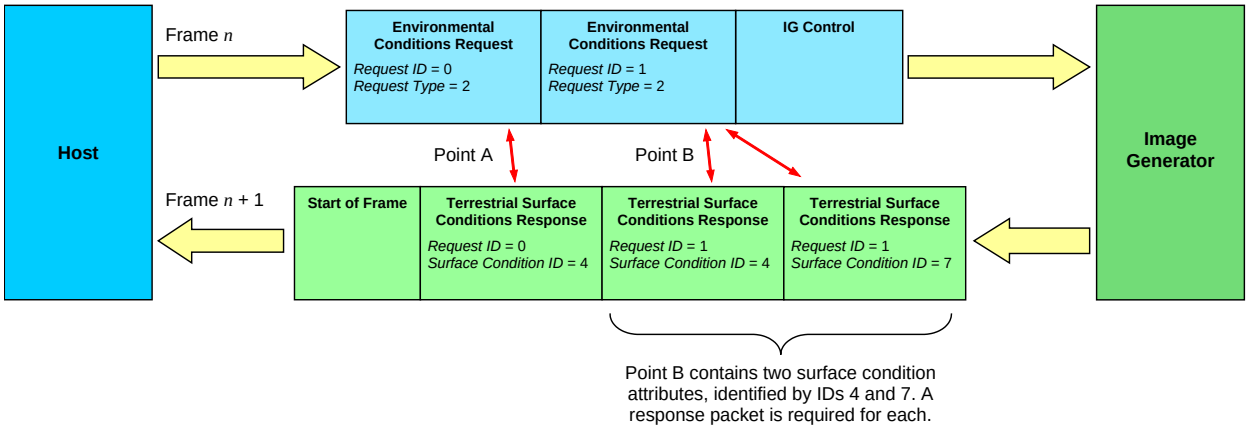


Figure 133 – Data Exchange for Terrestrial Surface Conditions Request

In the above example, the *Request ID* parameter of each **Environmental Conditions Request** packet is unique and is equal to the *Request ID* parameter in the corresponding **Terrestrial Surface Conditions Response** packet or packets. Because Point B falls within a region for which two surface condition attributes have been assigned, two separate response packets are required.

The contents of the **Terrestrial Surface Conditions Response** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 16																*Packet ID* = 0FF5h															
Request ID								Reserved																							
Surface Condition ID																															
Reserved																															

Figure 134 – Terrestrial Surface Conditions Response Packet Structure

Table 59 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 59 – Terrestrial Surface Conditions Response Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 16.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF5h	This parameter identifies this data packet as the **Terrestrial Surface Conditions Response** packet. The IG shall set this parameter to 0FF5h.
Request ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the environmental conditions request to which this response packet corresponds.
Surface Condition ID Type: unsigned int32 Units: N/A Values: 0 – 65,535	The IG shall set the value of this parameter to the presence of a specific surface condition or contaminant at the test point. Surface condition codes are IG-dependent.

6.2.13 Collision Detection Segment Notification

The **Collision Detection Segment Notification** packet is used to notify the Host when a collision occurs between a collision detection segment and a polygon. When a segment intersects a polygon whose material code matches the collision mask defined for the segment (see **Collision Detection Segment Definition** packet, Section 6.1.22), the IG sends a **Collision Detection Segment Notification** packet indicating where and with what the collision occurred. If a segment intersects multiple polygons with material codes matching the mask, only the closest intersection is returned. Segments are not tested against polygons belonging to the same the entity as the segment, and the IG shall not report such intersections.

Note that collision detection testing is performed every frame by the IG. If a collision detection segment has been disabled, it shall be excluded from all collision testing.

The contents of the **Collision Detection Segment Notification** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = 0FF4h															
Entity ID																*Contacted Entity ID*															
Segment ID								Reserved								*1	Reserved															
Material Code																															
Intersection Distance																															
Reserved																															

*1 *Collision Type*

Figure 135 – Collision Detection Segment Notification Packet Structure

Table 60 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 60 – Collision Detection Segment Notification Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF4h	This parameter identifies this data packet as the **Collision Detection Segment Notification** packet. The IG shall set this parameter to 0FF4h.
Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity that contains the collision detection segment.

Parameter	Description
Contacted Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity with which the collision occurred. The Host shall ignore this parameter if *Collision Type* is set to Non-entity (0).
Segment ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the ID of the collision detection segment along which the collision occurred. This parameter, along with *Entity ID*, allows the Host to match this response with the corresponding request.
Collision Type Type: 1-bit field Units: N/A Values: 0 Non-entity 1 Entity	The IG shall set the value of this parameter to indicate whether the collision occurred with another entity or with a non-entity object such as the terrain.
Material Code Type: unsigned int32 Units: N/A	The IG shall set the value of this parameter to the material code of the surface at the point of collision.
Intersection Distance Type: single float Units: meters Datum: *X1*, *Y1*, and *Z1* specified in **Collision Detection Segment Definition** packet	The IG shall set the value of this parameter to the distance along the collision test segment from the source endpoint (defined by the *X1*, *Y1*, and *Z1* parameters in the **Collision Detection Segment Definition** packet) to the point of intersection.

6.2.14 Collision Detection Volume Notification

The **Collision Detection Volume Notification** packet is used to notify the Host when a collision occurs between two collision detection volumes (see **Collision Detection Volume Definition** packet, Section 6.1.23). Volumes belonging to the same entity are not tested against each other, and the IG shall not report such intersections.

The IG shall send a **Collision Detection Volume Notification** packet for each volume involved in a collision. For instance, if two volumes collide, two **Collision Detection Volume Notification** packets shall be sent. If a collision occurs that involves three volumes, a total of six **Collision Detection Volume Notification** packets shall be sent.

Unlike with collision detection segment testing, where the result is a single point, the result of a collision detection volume test is the geometric intersection of two volumes. This intersection is usually an irregular volume with many vertices; therefore, the collision response data contains no spatial information describing the intersection.

Because collision detection volume testing does not involve polygon surfaces, no material code is returned with the collision response data.

Note that collision detection testing is performed every frame by the IG. If a collision detection volume has been disabled, it shall be excluded from all collision testing.

The contents of the **Collision Detection Volume Notification** packet are as follows:

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">Packet Size = 16</td><td colspan="16">Packet ID = 0FF3h</td></tr>
<tr><td colspan="16">Entity ID</td><td colspan="16">Contacted Entity ID</td></tr>
<tr><td colspan="8">Volume ID</td><td colspan="7">Reserved</td><td>*1</td><td colspan="8">Contacted Volume ID</td><td colspan="8">Reserved</td></tr>
<tr><td colspan="32">Reserved</td></tr>
</table>

*1 *Collision Type*

Figure 136 – Collision Detection Volume Notification Packet Structure

Table 61 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 61 – Collision Detection Volume Notification Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 16.

Parameter	Description
Packet ID Type: unsigned int16 Units: N/A Value: 0FF3h	This parameter identifies this data packet as the **Collision Detection Volume Notification** packet. The IG shall set this parameter to 0FF3h.
Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity that the collision detection volume belongs.
Contacted Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity with which the collision occurred. The Host shall ignore this parameter if *Collision Type* is set to Non-entity (0).
Volume ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the ID of the collision detection volume within which the collision occurred. This parameter, along with *Entity ID*, allows the Host to match this response with the corresponding request.
Collision Type Type: 1-bit field Units: N/A Values: 0 Non-entity 1 Entity	The IG shall set the value of this parameter to indicate whether the collision occurred with another entity or with a non-entity object such as the terrain.
Contacted Volume ID Type: unsigned int8 Units: N/A	The IG shall set the value of this parameter to the ID of the collision detection volume with which the collision occurred.

6.2.15 Animation Stop Notification

The **Animation Stop Notification** packet is used to indicate to the Host when an animation has played to the end of its animation sequence.

The IG shall not send a **Animation Stop Notification** packet if an animation is set to play continuously.

The contents of the **Animation Stop Notification** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 8																*Packet ID* = 0FF2h															
Entity ID																Reserved															

Figure 137 – Animation Stop Notification Packet Structure

Table 62 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 62 – Animation Stop Notification Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 8.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF2h	This parameter identifies this data packet as the **Animation Stop Notification** packet. The IG shall set this parameter to 0FF2h.
Entity ID Type: unsigned int16 Units: N/A	The IG shall set the value of this parameter to the entity whose animation has stopped.

6.2.16 Event Notification

The **Event Notification** packet is used to pass event data to the Host. The Host may enable and disable individual events using either the **Component Control** (Section 6.1.4) or **Short Component Control** (Section 6.1.5) packet.

This packet contains three user-defined 32-bit word values that may contain data describing the attributes of the event (i.e., time of occurrence, position). These data may be formatted as needed; however, they shall be byte-swapped as 32-bit fields when byte swapping is necessary. Refer to the description of the **Component Control** packet (Section 6.1.4) for more information.

The contents of the **Event Notification** packet are as follows:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Packet Size = 24																*Packet ID* = 0FF1h															
Event ID																Reserved															
Event Data 1																															
Event Data 2																															
Event Data 3																															
Reserved																															

Figure 138 – Event Notification Packet Structure

Table 63 defines each parameter's data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 63 – Event Notification Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The IG shall set this parameter to 24.
Packet ID Type: unsigned int16 Units: N/A Value: 0FF1h	This parameter identifies this data packet as the **Event Notification** packet. The IG shall set this parameter to 0FF1h.
Event ID Type: unsigned int16 Units: N/A	This parameter indicates which event has occurred. Event ID assignments are IG-specific.

Parameter	Description
Event Data 1 Type: word Units: Event-specific Values: Event-specific	This parameter is one of three 32-bit words used for user-defined event data. If this parameter is not needed to describe the event, this value is ignored. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple 8- or 16-bit values, the data should be packed so that byte swapping shall be performed correctly.
Event Data 2 Type: word Units: Event-specific Values: Event-specific	This parameter is one of three 32-bit words used for user-defined event data. If this parameter is not needed to describe the event, this value is ignored. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple 8- or 16-bit values, the data should be packed so that byte swapping shall be performed correctly.
Event Data 3 Type: word Units: Event-specific Values: Event-specific	This parameter is one of three 32-bit words used for user-defined event data. If this parameter is not needed to describe the event, this value is ignored. Note: This parameter shall be byte-swapped as a 32-bit value if the receiver and sender use different byte ordering schemes. If the parameter is used to store multiple 8- or 16-bit values, the data should be packed so that byte swapping shall be performed correctly.

6.2.17 Image Generator Message

The **Image Generator Message** packet is used to pass error, debugging, and other text messages to the Host. These messages may be saved to a log file and/or written to the console or other user interface. Because file and console I/O are not typically real-time in nature, it is recommended that the IG only send **Image Generator Message** packets while in Debug mode.

Each message is composed of multiple UTF-8 code points (characters). Each code point is represented by one to four bytes (octets). The text message shall be terminated by NULL, or zero (0). If the terminating byte is not the last byte of the eight-byte double-word, then the remainder of the double-word shall be padded with zeroes. Zero-length messages shall be terminated with four bytes containing NULL (to maintain 64-bit alignment). The maximum text length is dependent upon the sizes of the individual UTF-8 character data.

The *Packet Size* parameter shall contain the total number of bytes within the message, including the four-byte header, the message ID, the terminating NULL and any padding. This value shall be an even multiple of eight (8). For example, if the string, “Error 1234,” were sent to the Host, the packet size would be 24. Figure 139 illustrates the byte allocation for the packet:

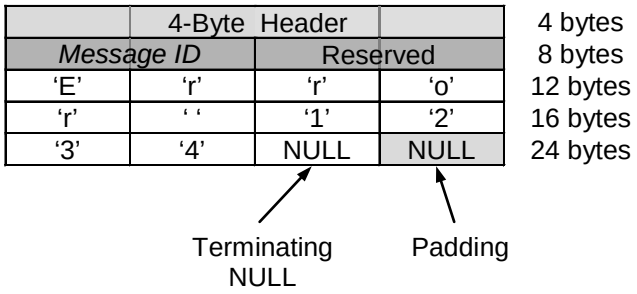


Figure 139 – Example of Image Generator Message Packet

The *Message ID* parameter identifies the message. Typically, this ID corresponds to a common, fixed message. In these cases, the Host may retrieve the message from a look-up table without the IG having to use unnecessary bandwidth to reproduce the text. The ID might also correspond to a common message while the remainder of the packet contains additional text to supplement the message.

The contents of the **Image Generator Message** packet are as follows:

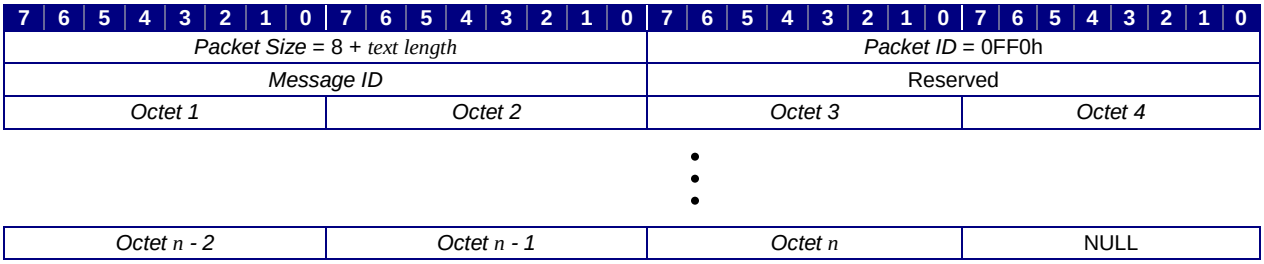


Figure 140 – Image Generator Message Packet Structure

Table 64 defines each parameter’s data type, units, and usage. If a default value and/or reference datum are applicable for a parameter, or if the domain differs from the range of values listed in Table 2, those values are also listed below.

Table 64 – Image Generator Message Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: bytes Value: 16 ≤ (8 + *text length*) ≤ 65528 where *text length* is the number of characters, including NULL characters, described by this packet	This parameter indicates the number of bytes in this data packet. The IG shall set this size using the formula in the left column. This value shall be divisible by eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 0FF0h	This parameter identifies this data packet as the **Image Generator Message** packet. The IG shall set this parameter to 0FF0h.
Message ID Type: unsigned int16 Units: N/A	This parameter specifies a numerical identifier for the message.
Octet n Type: octet Units: N/A	These 8-bit data are used to store the UTF-8 code points in the message string. Note: The maximum length of the string, including a terminating NULL, is 100 bytes.

6.3 Extension Packets

Extension data packets may be used to facilitate transmission of data not specifically supported by an existing CIGI standard packet. Extension data packets may be contained in both IG-to-Host and Host-to-IG messages. There are two kinds of extension packets: registered and locally defined. Registered extension packets shall be submitted to the CIGI Product Support Group (PSG) per Section 6.3.1 and upon acceptance shall be assigned a Packet ID within the range 1000h – 7FFFh.

A registered extension shall minimally define a label and the packet length, but submitters are encouraged to define the entire packet structure so SISO may consider it for adoption into a future version of the standard. If a variable size packet is desired, the submitter shall describe how the size is computed. The label shall be a compact and human-readable designator used to denote the extension. A commonly used registered extension could be selected for integration into a future version of the CIGI ICD as a standard packet, at which time the registered extension's Packet ID should be deprecated. CIGI compliance checking shall verify the existence of registered extension IDs and their respective packet size, but the remainder of each packet shall be ignored.

Locally defined extension Packet IDs shall be in the range 8000h - FFFEh. Users are encouraged to decrement from FFFEh to avoid straying outside the allowed range. CIGI compliance activities shall not validate locally defined extensions. Locally defined extensions are intended for experimentation only and shall not be used for a deliverable system.

When extension data packets are introduced into a particular CIGI application, they shall adhere to the standard data packet format in order to maintain continuity across data packets. Standard data packet format includes the *Packet Size* in the first two bytes and the *Packet ID* in the next two bytes. All parameters used to identify an instance of a packet (in other words, all parameters that, if identical, would allow two packets to represent the same object) should be contained within the first two 32-bit words. It is recommended that if bytes are allocated for such parameters as *Entity ID* and *View ID*, these values be positioned and sized in similar fashion as other instances of like information within the interface.

The size of each extension data packet depends on the amount of data contained in the packet. The developer shall place the total number of bytes, including the four bytes of header, in the *Packet Size* field of this packet so the receiver properly interprets the packet.

Note: All extension packets shall begin and end on a 64-bit boundary. The value of the *Packet Size* parameter shall therefore be an even multiple of eight (8).

Note that when an extension packet is introduced into a particular implementation of CIGI, that implementation no longer conforms to the baseline definition of the interface and thus may not be acceptable to the general CIGI user community. Each Host and IG device shall be capable of being configured to ignore all extension packets and should be capable of being configured to ignore specific registered extension packets.

The general format for extension packets is as follows:

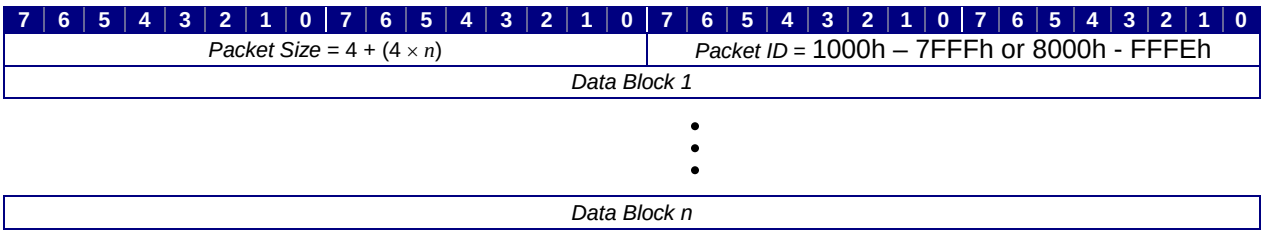


Figure 141 – General Extension Packet Structure

Table 65 defines each parameter's data type, units, and usage.

Table 65 – General Extension Packet Parameter Definitions

Parameter	Description
Packet Size Type: unsigned int16 Units: Bytes Value: $4 + (4 \times n) \geq 8$	This parameter indicates the number of bytes in this data packet. The value of this parameter shall be 4 times the number of data words, plus 4 for the word that includes the two-byte header. Note: Because all packets need to begin and end on a 64-bit boundary, the value of this parameter shall be an even multiple of eight (8).
Packet ID Type: unsigned int16 Units: N/A Value: 1000h – 7FFFh or 8000h - FFFEh	This parameter identifies this data packet as an extension packet. The value of this parameter shall be within the range of 1000h through 7FFFh, inclusive, for registered extensions and 8000h through FFFEh, inclusive, for locally defined extensions.
Data Block n Type: Packet-specific Units: Packet-specific	These 32-bit blocks contain extension data. The data may be arranged and formatted as needed.

6.3.1 Registration

This International Standard allows new CIGI extension packets to be specified by registration. These new packets are termed *registered extensions*.

New extension packets are registered using the established procedures of the registration authority for this International Standard (currently the CIGI PSG). These procedures require the submitter to supply all information for a new CIGI registered extension packet. Registration shall be according to the procedures in Section 6.3.2. Registration shall not be used to modify any existing standard packet or registered packet.

Other International Standards that normatively reference this International Standard, implementations of those standards, and implementations of this International Standard shall not use any CIGI registered packet IDs in the value ranges reserved for registration or future standardization by this International Standard with any meaning other than the one defined in this International Standard.

This clause specifies the rules and guidelines that shall be followed in preparing registration proposals. Registration proposals include required information for new CIGI registered extensions, as well as accompanying administrative information (see Appendix C).

6.3.2 Specification Elements for CIGI Registered Packets

6.3.2.1 Introduction

The specification of each CIGI registered extension shall include the following elements:

a) Label: a unique, compact, character string that is used to denote the registered extension,
b) Packet ID: a unique integer assigned by the CIGI PSG that is used to denote the registered item, and
c) Other concept-dependent information as required in this International Standard.

Other concept-dependent information may include the following elements:

a) A description of the extension,
b) A specification for each parameter defined in the *Data Blocks* and,
c) References.

6.3.2.2 Label

The label element of a CIGI registered extension specification shall be a compact and human-readable designator that is used to denote that registered packet. Labels in this International Standard may include the name or names for the registered item.

Each label in this International Standard shall:

a) Uniquely denote a specific packet,
b) Be a succinct expression of the concept instance that it denotes,
c) be represented as a character string, and
d) be human readable.

For presentation purposes only, a long label may be displayed on more than one line by using a hyphen (-) to separate the label before an underscore (_) character.

EXAMPLE The label DYNAMIC DEFORMABLE TERRAIN may be displayed for presentation purposes as:
DYNAMIC DEFORMABLE-
TERRAIN.

6.3.2.3 Description

The contents of the description element of a CIGI registered extension description shall be a precise statement of the nature, properties, scope, or essential qualities of the extension.

The descriptions of standard CIGI packets in this International Standard were created by applying a set of guidelines. Descriptions for proposed CIGI extensions shall be created according to these guidelines:

a) A description shall be provided for each CIGI extension. This description shall contain at least one word, number, expression or formula.
b) Data types should be those defined in Section 4.6 where applicable.
c) Descriptions shall be clear and concise, containing only the content necessary to summarize the extension.
d) Jargon shall not be used.
e) Abbreviations shall not be used.
f) Acronyms shall be used only if they are defined in Section 3.2 Acronyms and Abbreviations.

6.3.2.4 Specification

The contents of the specification elements of a CIGI registered extension specification shall be a precise statement of the nature, properties, scope, or essential qualities of the parameters defined in the packet's *Data Blocks*.

The specifications of standard CIGI packets in this International Standard were created by applying a set of guidelines. Specifications for proposed CIGI registered extensions shall be created according to these guidelines:

a) The extension packet structure shall conform to Section 4.5.
b) A description shall be provided for each CIGI extension packet parameter. This description shall contain at least one word, number, expression or formula.
c) Data types should be those defined in Section 4.6 where applicable.

d) Specifications shall be clear and concise, containing only the content necessary to summarize the extension packet data.
e) Jargon shall not be used.
f) Abbreviations shall not be used.
g) Acronyms shall be used only if they are defined in Section 3.2 Acronyms and Abbreviations.

6.3.3 References

6.3.3.1 Introduction

Two types of references are recognized in International Standards. The first type of reference is a normative reference [ISOD2]. Identified provisions of a normative reference are incorporated by reference and "become" part of the subject standard. Normative references play a key role in ensuring the consistency of the body of International Standards by allowing work done by others to be reused without modification. The second type of reference is an informative reference [ISOD2]. Identified provisions of an informative reference are cited as being the source of, related to, or providing additional information about text in the subject standard, but the identified provisions of the document are not themselves directly incorporated into the subject standard.

6.3.3.2 Citation Format

Each citation consists of an identifier and an optional location enclosed in square brackets ([ ]) with the identifier listed first, followed by a comma, followed by the location. The identifier specifies the cited document and shall appear in Section 2 References (Normative). The location specifies the portion of the document that is cited. Whenever possible, the location shall be specified in accordance with the requirements in [ISOD2]. When a cited document lacks a subclause structure, the location may be specified in a convenient and natural format depending on the organization of the cited document.

EXAMPLE [83502T, App. A-1, "HO"] and [RIIC, Table IV, "Saturn"].

Appendix A Bibliography (Informative)

H.J.P. Smith, D. D. (n.d.). *FASCODE Fact Sheet.* Retrieved September 23, 2013, from Kirtland Air Froce Base Library: http://www.kirtland.af.mil/library/factsheets/factsheet.asp?id=7913

ISO/IEC. (2009, 7 15). *SEDRIS SRM.* Retrieved 10 18, 2013, from SEDRIS.org: http://standards.sedris.org/#18026

ISO/IEC. (2013, 07 15). ISO/IEC 9973:2013(E). *Information technology - Computer graphics, image processing and envrionmental data representation - Procedures for registration of items*. ISO/IEC.

ISO/IEC. (2013, 3 8). *SEDRIS Standards*. Retrieved 10 18, 2013, from SEDRIS.org: http://standards.sedris.org/

SEDRIS. (2009, November 18). *SRM Orientation, Velocity & Acceleration Transformations V2.0.* Retrieved October 1, 2013, from SEDRIS.org: http://www.sedris.org/srm_4.4/src/lib/srm/docs/SRM_O&V_Transforms_Users_Manual_V20.pdf

Spectral Sciences, Inc. (2012). *MODTRAN5*. Retrieved September 23, 2013, from MODTRAN5: http://www.modtran5.com/index.html

Appendix B CIGI 4.X Change History

Version 4.0

General Changes

- Added support for 2D textured symbology; Problem/Change Request (PCR) 14 implementation.
- Restructured entity packets.
- Renamed several packets.
- Packet Size and Packet ID parameters expanded to 16-bits each; PCR 12 implementation.

Acceleration Control

- **Trajectory Definition** packet renamed.
- Added angular acceleration parameters.
- Added coordinate system articulated part parameters similar to **Velocity Control**.
- *Retardation Rate* and *Terminal Velocity* parameters removed.

Animation Control

- New packet implementing PCR 15.

Appendices

- Appendix A – Changed to Informative Bibliography.
- Added new Appendix C – Extension Registration.
- Acronyms Appendix – Moved to Section 3.2
- Errata Appendix removed.

Celestial Sphere Control

- Added *Seconds* parameter; implements PCR 13b.
- Renamed *Ephemeris Model Enable* to *Continuous Time-of-Day Enable*.

Conformal Clamped Entity Position

- **Conformal Clamped Entity Control** packet renamed.

Entity Control

- Subset of previous packet by this name containing parameters expected to change infrequently.

Entity Position

- Subset of previous **Entity Control** packet containing parameters for entity position and attitude.

Extension Packets

- Replaces the User-Defined Packets from previous CIGI versions.
- Defines an extension registration process.
- Utilizes the expanded Packet ID architecture provided by the implementation of PCR 12.

Symbol Polygon Definition

- **Symbol Line Definition** packet renamed.

Symbol Textured Circle Definition

- New packet implementing PCR 14.

Symbol Textured Polygon Definition

- New packet implementing PCR 14.

Velocity Control

- **Rate Control** packet renamed.

Weather Control

- Scud parameters expanded to control top and bottom independently; implements PCR 18.

Appendix C Extension Registration

C.1 Introduction

This appendix contains a blank form that may be used to create proposals for CIGI registered items as specified in Section 6.3.

All forms include four common initial elements that apply to all types of registerable items. These common elements are described in Table 66.

Table 66 - COMMON REGISTRATION PROPOSAL SPECIFICATION ELEMENTS

Element	Definition
Date of proposal	Date the registration proposal is submitted.
Sponsoring authority	Sponsor of the registration proposal.
Justification for inclusion	Justification for the registration proposal.
Additional comments	Any additional comments on the registration proposal.

C.2 Proposal for the registration of an extension packet

Each proposal for the registration of an Extension Packet shall provide values of all of the elements contained in Table 66 and in Table 67. The required contents of the specification column are described in Section 6.3.2.

Table 67 – EXTENSTION REGISTRATION PROPOSAL ELEMENTS

Element	Specification
Description	
Extension label	
Packet Length	
Notes	
References	
Parameter 1	
Parameter n	Repeat as needed for each parameter.