Interface Control Document
for the
Common Image Generator Interface (CIGI)

Version 3.3

cigi.sourceforge.net

Document No: TST08I016
Issue Date: Nov. 3, 2008

Durham, Lance
The Boeing Company

Phelps, Bill
The Boeing Company

LIST OF PAGES

Title Page
ii through xii
1 through 267

USE AND DISCLOSURE OF DATA

This document is intended to be a public domain document. As such it does not contain any proprietary notices. The Boeing Company may make improvements and changes to the contents of this document at any time without notice. The Boeing Company assumes no responsibility for the use of the information in this document. This document may contain technical inaccuracies or typographical errors. Periodic changes are made to the information contained herein. These changes will be incorporated in new editions of the document.

To maintain configuration control of the interface it is requested that this document not be reproduced in whole or in part for the purpose of modification to the interface. Please forward all modification suggestions to the maintainer shown at the bottom of the cover page for discussion and possible incorporation into the interface.

TABLE OF CONTENTS

TABLE OF FIGURES

Figure Page

TABLE OF TABLES

Tables Page

INDEX OF CHANGED PAGES

Revision	Page(s) Affected*	Remarks	Revised By
3.0		New document.	L. Durham
3.1	44	The caption of Figure 28 has been changed to "Conformal Clamped Entity Control Packet Structure."	L. Durham
	67-68	The reference data listed in Table 13 for the linear and angular rates of entities were incorrectly given as "Entity coordinate system." These have been changed to "Geodetic reference plane."	L. Durham
	81-82	Added examples of using environmental regions to create gridded or tiled weather cells.	L. Durham
	86	The range of valid values for the *Transition Perimeter* parameter has been changed to ≥0.	L. Durham
	94-95	The parameters in Table 19 have been rearranged to reflect the order in which the data appear in the Maritime Surface Conditions Control packet.	L. Durham
	123	The reference data for *Acceleration X*, *Acceleration Y*, and *Acceleration Z* parameters have been defined as the ellipsoid-tangential NED reference coordinate system.	L. Durham
	123	The units of the *Retardation Rate* parameter have been changed to m/s^2. The description for this parameter has been clarified.	L. Durham
	158	The 4th byte of the first word of the Environmental Conditions Request packet was incorrectly designated as Reserved. This parameter is now correctly labeled as *Request ID* and a description for this parameter has been added to Table 34.	L. Durham
	177-180	The use of the *Gate X Position* and *Gate Y Position* parameters has been clarified.	L. Durham
	179	The caption of Figure 88 has been changed to "Sensor Response Packet Structure."	L. Durham
	179	The caption of Table 40 has been changed to "Sensor Response Parameter Definitions."	L. Durham
	183	The use of the *Gate X Position* and *Gate Y Position* parameters has been clarified.	L. Durham
	197	The range of the *Surface Conditions ID* parameter in Table 46 has been defined.	L. Durham
	205	Figure 103 has been corrected to show the *Message ID* parameter.	L. Durham
3.2	various	Trivial typographical, grammatical, or other minor corrections or revisions.	L. Durham
	1	Swapped and Revised Sections 1.1 and 1.2. Revised Section 1.3 to include minor version numbers and to remove restrictions governing changes between versions.	L. Durham
	4	Added Section 2.1.2 to contain text on the use of the frame numbers.	L. Durham
	5–6, 32, 167	Changed the use of the *Frame Counter* parameters of the Start of Frame and IG Control packets. Renamed the parameters to "Host Frame Number" and "IG Frame Number."	L. Durham
	11	Modified Figure 10 to reflect the new use of the frame numbers.	L. Durham
	13	Figure 11 has been updated so that the arrangement and order of bit fields within each byte reflect the way bit fields are contained in actual data packets.	L. Durham
	14	Section 2.5 has been revised so that User-Defined packets do not imply non-compliance and that a device must gracefully accept unrecognized packets.	L. Durham
	15	Added description and use of minor version. Inserted a new rule as Rule #3.	L. Durham
	19	Figure 14 has been modified to accurately show the left and right field-of-view half-angles.	L. Durham
	27	Figure 22 has been updated to match the changes in Section 4.2.3.	L. Durham
	31–33	Renamed the *CIGI Version* parameter of the IG Control packet to "Major Version." Added the *Minor Version* parameter to the packet. Renamed *Byte Swap* parameter to "Byte Swap Magic Number."	L. Durham
	33, 168	Added a reference datum to the *Timestamp* parameter of both the IG Control and Start of Frame packets.	L. Durham
	36	The behavior of animations has been clarified for the Stop state of the *Animation State* parameter. Table 5 has been corrected to show the proper behavior for this state.	L. Durham
	37, 39	Changed the name of *Collision Detection Request* flag to *Collision Detection Enable* in the Entity Control packet.	L. Durham
	41	The behavior of the Stop state for the *Animation State* parameter has been clarified in Table 7. The description for the *Entity Type* parameter has also been modified to address changing of an existing entity's type.	L. Durham
	42-43	The descriptions for *Yaw*, *Pitch*, and *Roll* rotations for child entities have been corrected/clarified.	L. Durham
	51	Fixed the order of the Component Control packet's header fields in Figure 32.	L. Durham
	67–70	Added a *Coordinate System* parameter to the Rate Control packet. Added a description of this parameter to Table 13. Modified reference data for linear and angular rates for entities.	L. Durham
	90	Specified weather behavior for overlapping weather layers according to Table 16.	L. Durham
	111	Removed the restriction in the second paragraph that each sensor is associated with exactly one view. This allows sensor data to be displayed on multiple displays.	L. Durham
	118	Added a sentence specifying that if multiple head trackers are assigned for a given view or viewport, then the IG determines how those inputs are combined.	L. Durham
	125	Corrected descriptions of *Acceleration Y* and *Acceleration Z* parameters in Table 26. Specified a reference datum for *Retardation Rate* parameter in Table 26.	L. Durham
	141–156	Added the *Update Period* parameter to the HAT/HOT Request, Line of Sight Segment Request, and Line of Sight Vector Request packets. Renamed *Byte Swap* parameter to "Byte Swap Magic Number."	L. Durham
	144–150	Added *Destination Entity ID Valid* and *Destination Entity ID* parameters to the Line of Sight Segment Request packet. Also changed the name of *Entity ID* to *Source Entity ID*.	L. Durham
	164,167	Renamed the *CIGI Version* parameter of the Start of Frame packet to "Major Version." Added the *Minor Version* parameter to the packet.	L. Durham

Revision	Page(s) Affected*	Remarks	Revised By
3.2	169–183	Added the *Frame Counter LSN* parameter to the HAT/HOT Response, HAT/HOT Extended Response, Line of Sight Response, and Line of Sight Extended Response packets.	L. Durham
	184–189	Renamed *Frame Counter* parameter of the Sensor Response and Sensor Extended Response packets to "Host Frame Number." Modified the use of these parameters so that they contain the Host-to-IG frame counter.	L. Durham
	Appendix II	Reworked list of changes to include only changes from CIGI 3.0 and higher.	L. Durham
3.3	various	Trivial typographical, grammatical, or other minor corrections or revisions.	L. Durham
	xi–xii	Changed page numbering and figure and table references in this index to reflect the current numbering.	L. Durham
	1	Renamed Section 1.3 and inserted Section 1.4.	L. Durham
	3–15	Removed "2.1 Message Protocol" and "2.2 Data Packaging" headings and promoted and renumbered their subsections.	L. Durham
	3	Inserted Section 2.1.	L. Durham
	7	Clarified Figures 7 and 8 to emphasize the positions of the most significant bits. Indicated these bits with a highlight and added a sentence to the preceding paragraph to explain the highlight. Changed the "+/-" labels to "-2^{n-1}" to indicate two's complement representation.	L. Durham
	8	Modified the list of advantages related to byte swapping responsibilities.	L. Durham
	8–9	Clarified the packet ordering restrictions for entities, regions, symbols, symbol surfaces, and other object.	L. Durham
	9–10	Added Symbol Surface Definition, Symbol Text Definition, Symbol Circle Definition, Symbol Line Definition, Symbol Clone, Symbol Control, and Short Symbol Control packets to Table 1.	L. Durham
	12	Changed representation for alphanumeric data from ANSI to UTF-8 and added rationale for the change. Changed Table 2 to reflect unsigned range of a char datum (octet).	L. Durham
	12, 52, 212	Renamed "char" data type to "octet."	L. Durham
	13	Inserted a new Section 2.7.3.	L. Durham
	13	Renamed bit fields "Field 2," "Field 3," and "Field 5" to "F2," "F3," and "F5" in Figure 11 for consistency.	L. Durham
	14	Changed "should" to "must" in rule #2.	L. Durham
	15	Changed "opcode" to "packet ID" in rule #3.	
	18	Added a paragraph to Section 3.2.1 to introduce perspective and orthographic parallel view projections.	L. Durham
	22–23	Added Section 3.3.	L. Durham
	24	Revised Figure 17 for clarity.	L. Durham
	26	Revised Figure 19 for clarity.	L. Durham
	27	Changed the heading of Section 3.4.2 (formerly Section 3.3.2) to "Entity Coordinate System" for consistency.	L. Durham
	28	The Figure 21 and text have been clarified to emphasize that a submodel coordinate system may be arbitrarily rotated within the model.	L. Durham
	29–37	Added Sections 3.4.4 and 3.4.5.	L. Durham
	42–44	Added an *Extrapolation/Interpolation Enable* parameter to the IG Control packet.	L. Durham
	48–52	Added a *Linear Extrapolation/Interpolation Enable* parameter to the Entity Control packet.	L. Durham
	83	Changed the reference from local time to UTC for the *Hour* and *Minute* parameters.	L. Durham
	137	Modified first paragraph to clarify the IG's behavior when multiple View Definition packets are received.	L. Durham
	165	Clarified the reference datum and direction for the *Elevation* parameter of the Line of Sight Vector Request packet.	L. Durham
	175–211	Added Sections 4.1.29 through 4.1.35.	L. Durham
	219	Added Figure 111.	L. Durham
	221	Clarified the descriptions of the *Normal Vector Azimuth* and *Normal Vector Elevation* parameters.	L. Durham
	219–221	Removed references to deprecated Intersection Point Coordinate System parameter in the Line of Sight Extended Response packet.	L. Durham
	226	Corrected the paragraph describing the usage of the *Host Frame Number LSN* parameter.	L. Durham
	231	Clarified the descriptions of the *Normal Vector Azimuth* and *Normal Vector Elevation* parameters.	L. Durham
	241–242	Clarified the reference datum for *Yaw*, *Pitch*, and *Roll* parameters of the Position Response packet for submodel coordinate systems.	L. Durham
	259–260	Changed format of character data from 8-bit ANSI to multi-byte UTF-8. Changed names of character fields from "Character n" to "Octet n."	L. Durham

* As numbered in the listed revision.

1. INTRODUCTION

1.1 Purpose

This Interface Control Document (ICD) describes Version 3.3 of the open-source Common Image Generator Interface (CIGI). This ICD has been written for software engineers involved in the integration of a host simulator with an image generator (IG). This document contains descriptions of all data parameters, event sequences, and input/output (I/O) protocols necessary to accomplish this task using CIGI.

1.2 Scope

This document defines a common Host-to-IG interface standard to promote interoperability among Image Generators. This document is not intended to define IG functional requirements. Therefore, implementation of 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. 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 must be preserved but may be supplemented with user-defined packets.

1.3 Version Numbering

The CIGI version number contains a major and a minor revision number. The major version number is contained within the **IG Control** and **Start of Frame** data packets. This version number will always correspond to the major version number of the appropriate CIGI Interface Control Document.

A minor version number is also contained within the **IG Control** and **Start of Frame** data packets. This number will correspond to the minor version number of the CIGI ICD. Incrementing the minor version number will not affect compatibility with existing versions of the interface with the same major version number. In other words, the changes made to a minor revision will be designed as to be transparent to existing devices that use the same major version of CIGI but a prior minor version.

Note that a device may, in some cases, need to be aware that its counterpart uses an earlier minor version of CIGI. The reverse will not be true.

1.4 Instructions for Revising this Document

This document is maintained by the CIGI Standing Study Group (SSG) through the Simulation Interoperability Standards Organization (http://www.sisostds.org). All changes and revisions to the interface and this ICD will be implemented through the SSG.

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 (e.g., 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. 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 should 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 simplest scenario, a Host will communicate with exactly one IG as illustrated in Figure 1 and Figure 2. The physical connection may be made using an Ethernet hub or switch, or using a single crossover cable (i.e. a patch cable with the Transmit and Receive lines crossed).

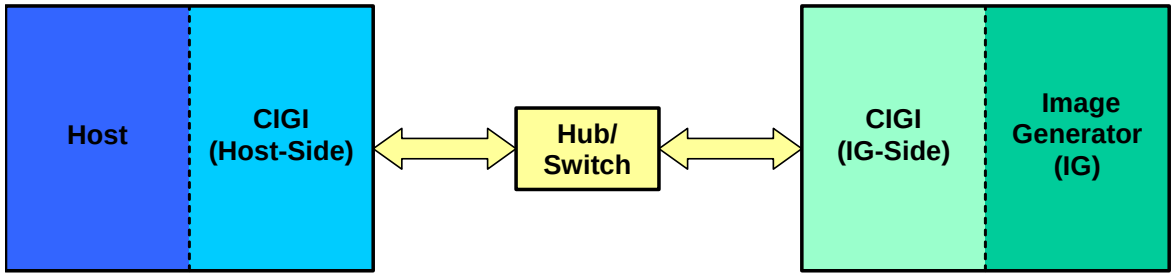


Figure 1 – Connection Using Ethernet Hub/Switch

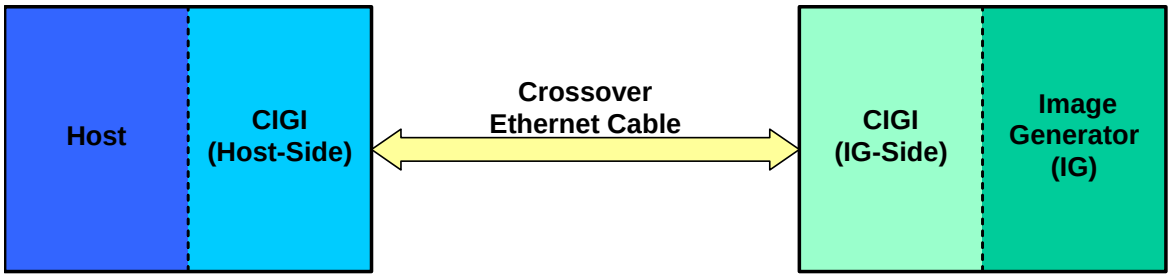


Figure 2 – Connection Using Crossover Cable

While it is suggested that every CIGI session (see Section 2.1) 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. Since CIGI relies on the underlying protocol to perform multiplexing and demultiplexing of messages across a physical connection, each Host device must communicate to the IG through a separate session. Further, all mission function and response data would only be valid for one Host. Finally, for each IG there can only be one Ownship (see Section 3.1.1).

The use of multiple sessions is generally preferable to using multiple receivers in the same session. This way the Host can store a separate state for each IG and all IGs can communicate its operational state, mission function responses, and other data to the Host.

CIGI sessions are described in the following section.

2.1 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 must 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 the reasons described in the preceding section.

The Host and IG must 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 4.2.1 and 4.1.1). 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.

2.2 Message Synchronization

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

2.2.1 Asynchronous Operation

During asynchronous operation, the Host sends data to the IG at a predetermined rate. 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 messages is determined by the host itself.

Meanwhile, the IG maintains its own frame rate, which is typically bound by the vertical sync signal from the display system. During each frame, the IG first checks its buffer for incoming CIGI messages. It then updates the scene graph based on the contents of the CIGI message along with any previously defined rates and trajectories, dynamic environmental attributes, and other factors. Finally, the IG renders the scene.

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 3.

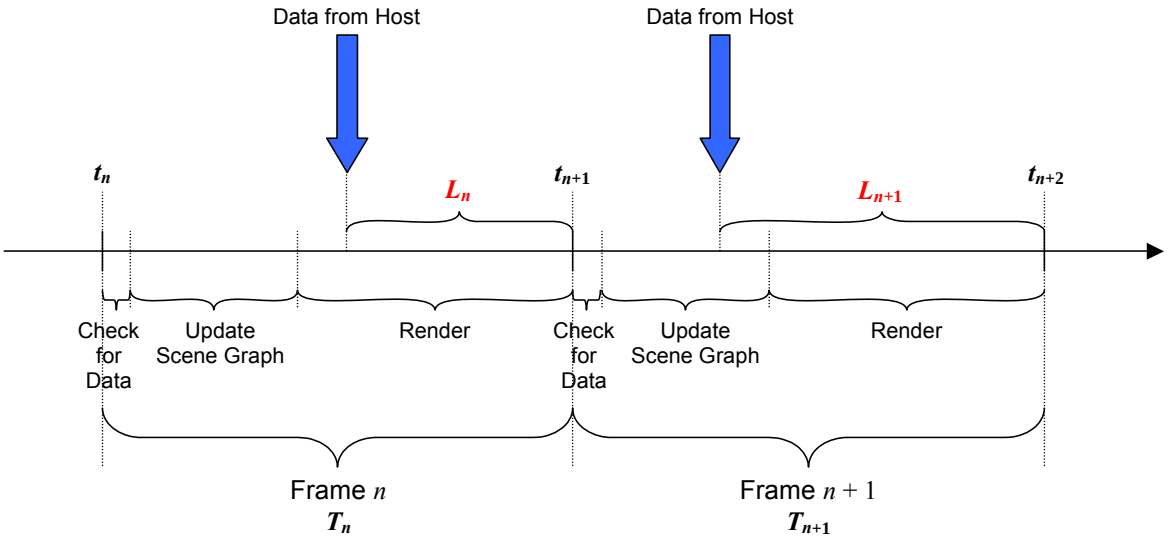


Figure 3 – Latencies Caused by Asynchronous Operation

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 old 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 4.1.1 and 4.2.1, respectively) include a *Timestamp* parameter that the IG can 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 **Rate Control** and **Trajectory Definition** packets (see Sections 4.1.8 and 4.1.20), the IG can calculate the probable positions of those entities during the current frame.

Because entities can be extremely dynamic, such extrapolations are prone to error. Asynchronous operation, therefore, is recommended only in low-fidelity applications or when synchronous operation is not possible.

2.2.2 Synchronous Operation

During synchronous operation, the IG sends a start-of-frame (SOF) message to the Host to signal the beginning of each frame. This message, which is usually driven by a vertical sync signal from the display system, functions as a "heartbeat" that dictates the timing of data transfers between the IG and Host. The SOF message also contains mission function responses, event notifications, and other IG data.

The Host immediately responds to each SOF 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 then begins 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 3. Also, because the Host's frame rate is regulated by the IG, this prevents jitter caused by the creeping effect of misaligned frames.

Figure 4 illustrates the sequence described above:

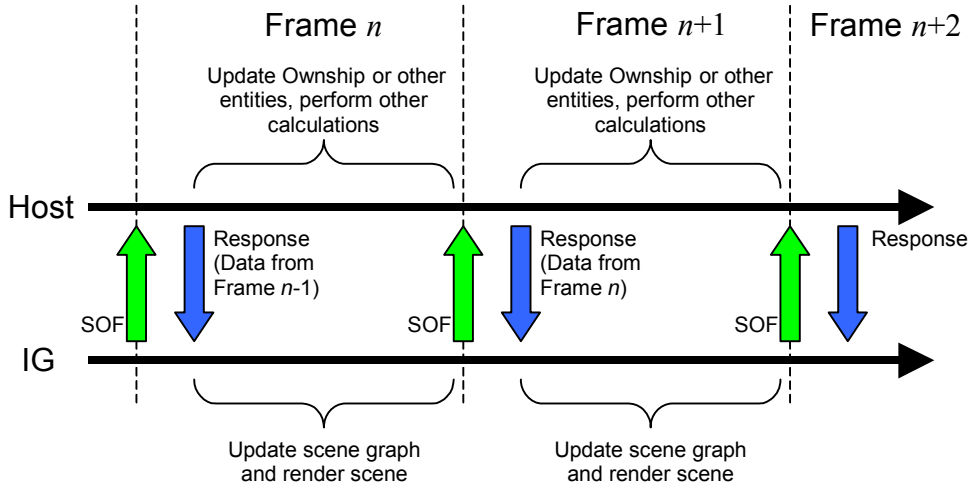


Figure 4 – Synchronous Start-of-Frame/Response Cycle

Depending upon bandwidth limitations and the transport delay, the IG may 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 can be introduced so that the IG sends each SOF 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 may vary from frame to frame, this offset can be adjusted to allow for worst-case network loads. Figure 5 illustrates the start-of-frame offset technique:

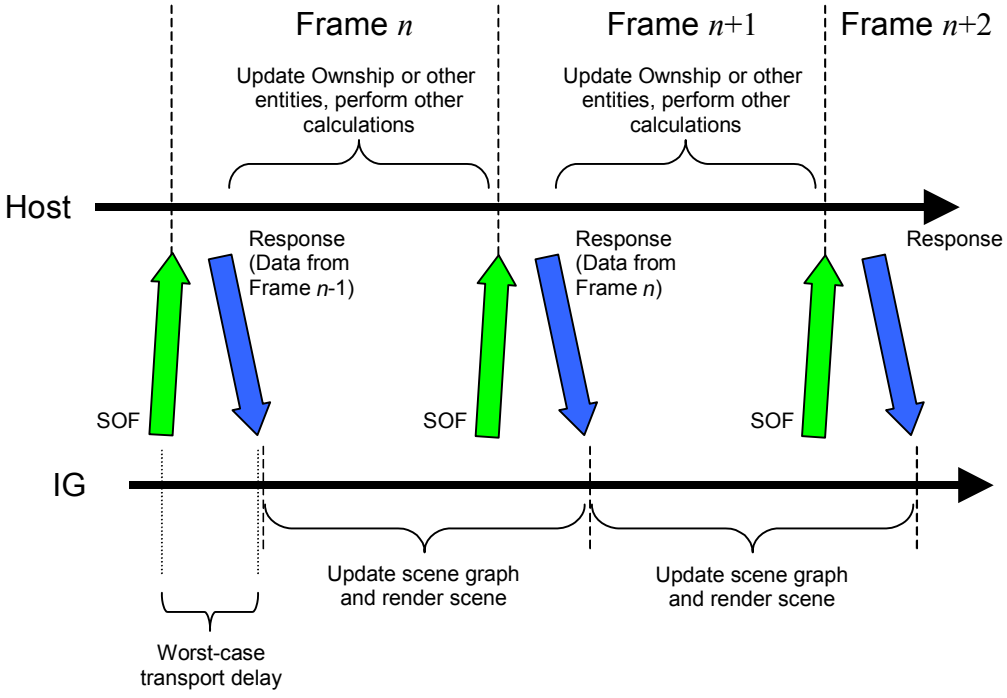


Figure 5 – Synchronous Message Timing Offset

2.3 Frame Numbering

To enable tracking of Ethernet messages, both the SOF message and the Host response message are tagged with sequential frame numbers. The frame number for each device is independent. Before sending its next message, the Host device should 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 should place this same value in the *Last Host Frame Number* parameter of the next outgoing **Start of Frame** packet.

Likewise, the IG device should increment its frame number value and populate the *IG Frame Number* parameter of the outgoing **Start of Frame** data packet, and the Host should 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:

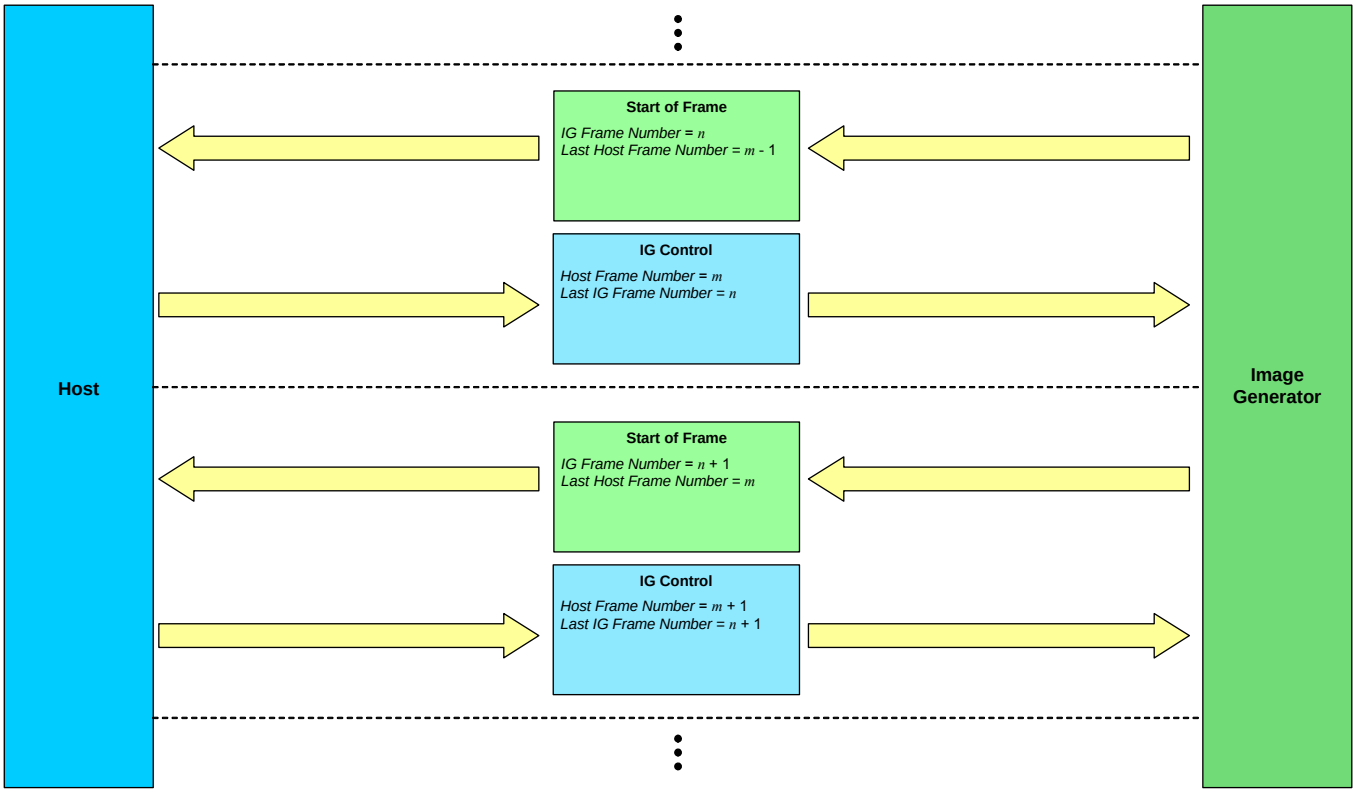


Figure 6 – Frame Numbering Sequence

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.” The *Frame Counter* values were not independent. 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.

2.4 Ethernet Message Frequency

Although the IG software can be configured to run at any reasonable frequency, the period is typically bound to some multiple of the display refresh rate. This makes the data frame rate very stable, and it alleviates tearing and other undesirable video anomalies. Common update rates are 30, 50, and 60 Hz, depending upon locale.

For synchronous operation, this means that the Host must run at a multiple of the display update rate or must extrapolate its data each frame to meet the specified IG update rate.

2.5 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. In a “big-endian” configuration, 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 7 illustrates the byte layout of signed and unsigned 16- and 32-bit data types on big-endian architectures. The most significant bits are highlighted.

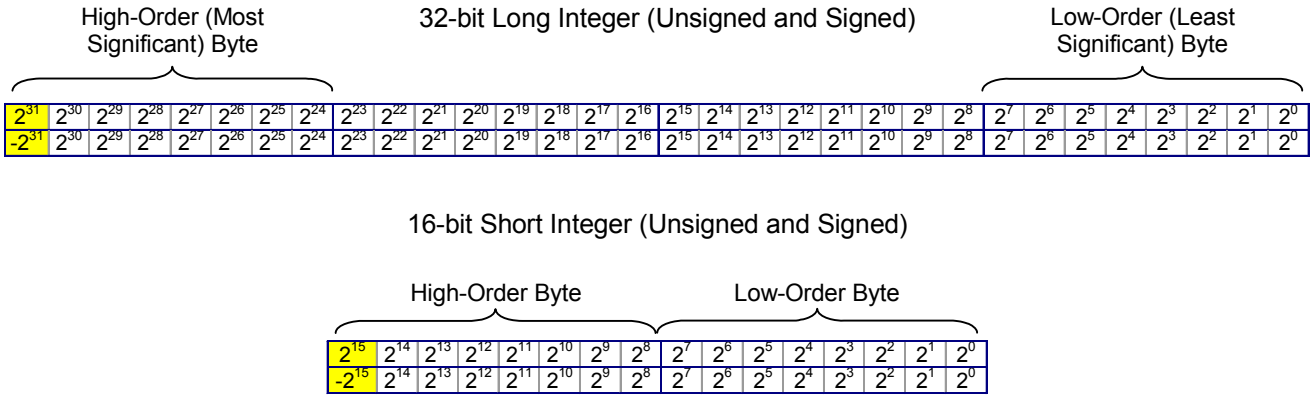


Figure 7 – Big-Endian Byte Order

Figure 8 illustrates the byte layout of the same data types on little-endian architectures:

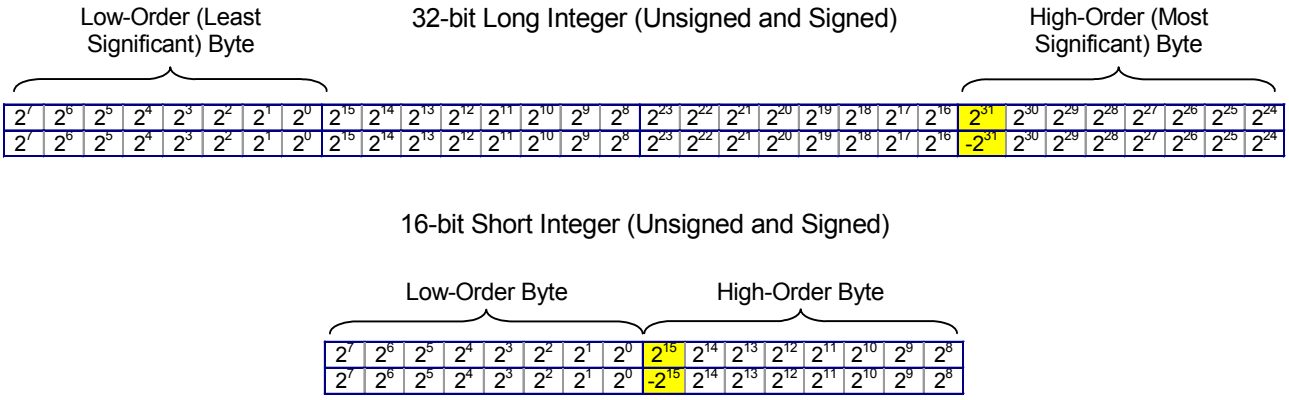


Figure 8 – Little-Endian Byte Order

CIGI 3 does not impose a byte order upon either the IG or the Host. Instead, the receiver has the responsibility of performing 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 Page 2). 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.

The **IG Control** and **Start of Frame** packets both contain a 16-bit *Byte Swap Magic Number* parameter that indicates whether the receiver should byte-swap the incoming message. The sender sets the most significant bit of the high byte and clears all other bits. When the receiver examines this parameter, it must byte-swap the entire message if the most significant bit of the *low* byte is set. Figure 9 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}			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	⇨	8000h	**No Swap**
0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	⇨	80h	**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}			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	⇨	80h	**Swap**
0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	⇨	8000h	**No Swap**

Figure 9 – Use of CIGI Byte Swap Magic Number Parameter

Refer to Sections 4.1.1 and 4.2.1 for additional details on the **IG Control** and **Start of Frame** packets.

2.6 Message Structure

CIGI messages are comprised of one or more data packets. During synchronous operation, there is exactly one IG-to-Host (SOF) message and one Host-to-IG response message per frame.

The first two bytes of each packet are the packet header. The first byte of this header contains an opcode that uniquely identifies the packet type; the second byte contains the size in bytes of the packet. 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 (e.g., a moving model or a view) are 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 must begin and end on 64-bit boundaries. Packets will be padded as necessary so that the packet size will be an even multiple of eight (8) bytes.

The first data packet in each message from the IG to the Host must be a **Start of Frame** packet (see Section 4.2.1). The message may also contain other IG-to-Host packets as listed in Table 1.

The first packet in each response message from the Host to the IG must be an **IG Control** packet (see Section 4.1.1). Zero or more Host-to-IG packets (see Table 1) may follow within the message.

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

Similarly, objects that have been deleted can 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 no other packet occurring later in that message or in a subsequent message can reference Entity *n* until a new Entity *n* is created.

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 frame, the Host should not send an **Entity Control** packet.

Table 1 – Data Packet Summary

Opcode	Data Packet Name	Mandatory Each Frame	Section
	HOST TO IG		
1	**IG Control**	Yes	4.1.1
2	**Entity Control**	No	4.1.2
3	**Conformal Clamped Entity Control**	No	4.2.3
4	**Component Control**	No	4.1.4
5	**Short Component Control**	No	4.1.5
6	**Articulated Part Control**	No	4.1.6
7	**Short Articulated Part Control**	No	4.1.7
8	**Rate Control**	No	4.1.8
9	**Celestial Sphere Control**	No	4.1.9
10	**Atmosphere Control**	No	4.1.10
11	**Environmental Region Control**	No	4.1.11
12	**Weather Control**	No	4.1.12
13	**Maritime Surface Conditions Control**	No	4.1.13
14	**Wave Control**	No	4.1.14
15	**Terrestrial Surface Conditions Control**	No	4.1.15
16	**View Control**	No	4.1.16
17	**Sensor Control**	No	4.1.17
18	**Motion Tracker Control**	No	4.1.18
19	**Earth Reference Model Definition**	No	4.1.19
20	**Trajectory Definition**	No	4.1.20
21	**View Definition**	No	4.1.21
22	**Collision Detection Segment Definition**	No	4.1.22
23	**Collision Detection Volume Definition**	No	4.1.23
24	**HAT/HOT Request**	No	4.1.24
25	**Line of Sight Segment Request**	No	4.1.25
26	**Line of Sight Vector Request**	No	4.1.26
27	**Position Request**	No	4.1.27
28	**Environmental Conditions Request**	No	4.1.28
29	**Symbol Surface Definition**	No	4.1.29
30	**Symbol Text Definition**	No	4.1.30
31	**Symbol Circle Definition**	No	4.1.31
32	**Symbol Line Definition**	No	4.1.32
33	**Symbol Clone**	No	4.1.33
34	**Symbol Control**	No	4.1.34
35	**Short Symbol Control**	No	4.1.35

Opcode	Data Packet Name	Mandatory Each Frame	Section
	IG TO HOST		
101	**Start of Frame**	Yes	4.2.1
102	**HAT/HOT Response**	No	4.2.2
103	**HAT/HOT Extended Response**	No	4.2.3
104	**Line of Sight Response**	No	4.2.4
105	**Line of Sight Extended Response**	No	4.2.5
106	**Sensor Response**	See Packet Description	4.2.6
107	**Sensor Extended Response**	See Packet Description	4.2.7
108	**Position Response**	No	4.2.8
109	**Weather Conditions Response**	No	4.2.9
110	**Aerosol Concentration Response**	No	4.2.10
111	**Maritime Surface Conditions Response**	No	4.2.11
112	**Terrestrial Surface Conditions Response**	No	4.2.12
113	**Collision Detection Segment Notification**	No	4.2.13
114	**Collision Detection Volume Notification**	No	4.2.14
115	**Animation Stop Notification**	No	4.2.15
116	**Event Notification**	No	4.2.16
117	**Image Generator Message**	No	4.2.17
	USER-DEFINED DATA PACKETS		
201 – 255	**User-Defined Data Packets**	Application-Dependent	4.3

Only a subset of these data packets are typically required in any given message to describe data changes to the IG. Figure 10 shows a hypothetical sequence of Host-to-IG messages. Note that each message begins with a **Start of Frame** or **IG Control** packet.

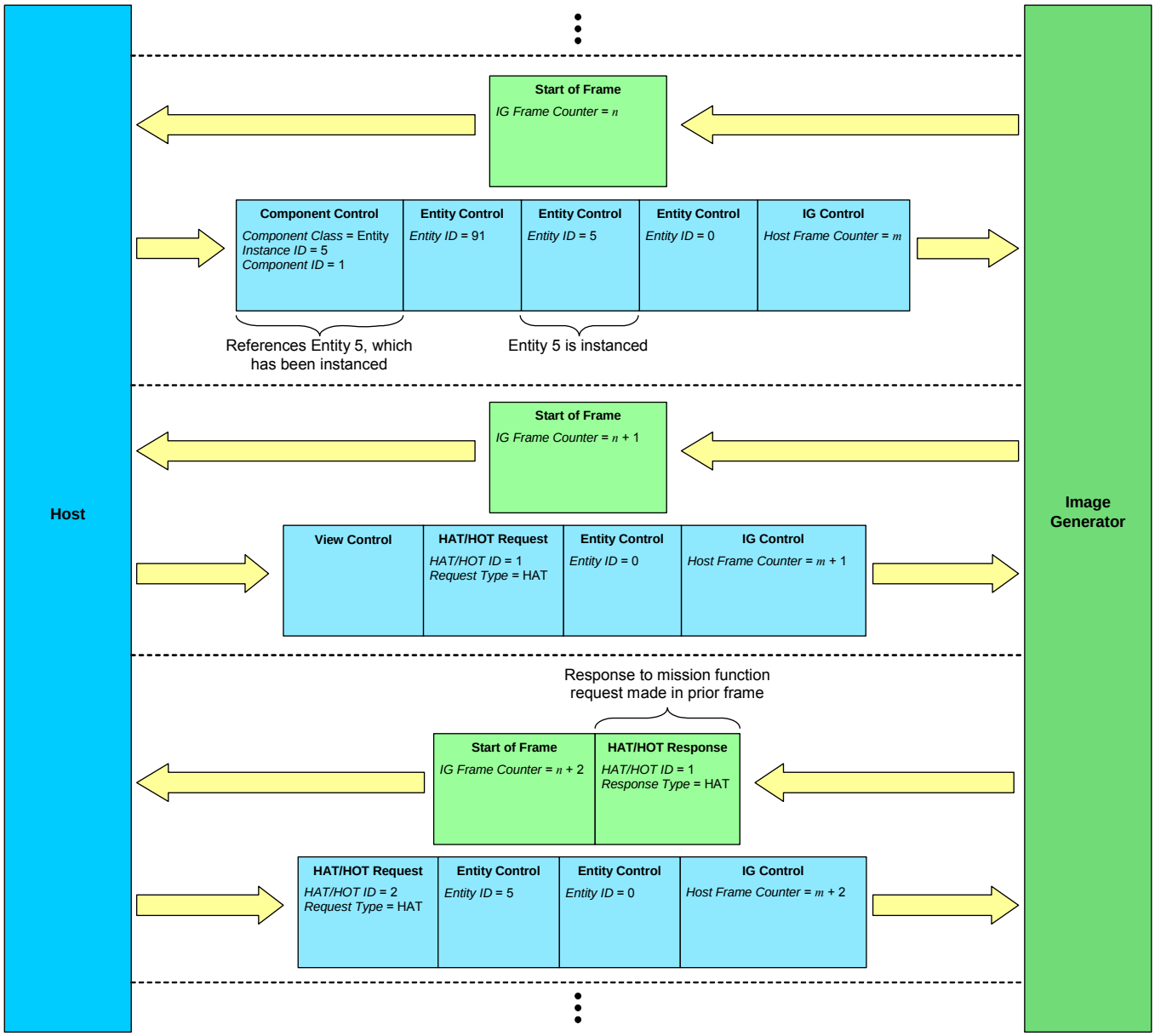


Figure 10 – Example of Message Exchange in Synchronous Mode

2.7 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.

All data are stored as two's complement binary values. If a Host or IG uses an alternate representation, that device must convert all data it writes to and reads from the interface.

2.7.1 Integral Types

Real values are stored 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.

Integer values are stored 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 bit fields as described in Section 2.7.3.

Alphanumeric characters are stored using from one to four octet (byte) fields as described in Section 2.7.2.

Generic values with no explicitly defined type are indicated as 32-bit **word** fields. Word fields may be used as the developer sees fit; however, they will be treated by the interface as unsigned int32 data. Any arithmetic operations performed on word data will be carried out as if on unsigned 32-bit integers. In addition, if byte swapping is necessary, word fields will be byte-swapped as 32-bit integers.

Table 2 – Integral 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

2.7.2 UTF-8 Strings

Alphanumeric characters are represented as Unicode code points using the UTF-8 encoding scheme. These code points are stored using from one to four octets. 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 the Unicode Standard web site (http://unicode.org).

CIGI implementers must 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 must ensure that such code points are not split when truncating a UTF-8 string.

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

2.7.3 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 bit 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 bit fields:

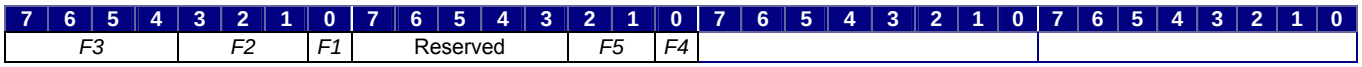


Figure 11 – Example of Multiple Bit Fields

The above word can be broken down as follows:

- *F1* is a single-bit binary value with possible values 0 and 1.
- *F2* is a 3-bit unsigned integer value with a range of 0 through $2^3 - 1$.
- *F3* is a 4-bit unsigned integer value with a range of 0 through $2^4 - 1$.
- *F4* is a single-bit binary value with possible values 0 and 1.
- *F5* is a 2-bit unsigned integer value with a range of 0 through $2^2 - 1$.
- The bits marked as “Reserved” are unused and must be set to zero by the sender.
- The remaining two bytes in this example are used to store two 8-bit values.

CIGI 3 arranges all bit fields so that they do not cross byte boundaries.

2.8 Startup and Shutdown Sequences

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

While the IG initializes, it should disregard any CIGI messages from the Host. During synchronous operation, the Host should communicate with the IG only in response to a start-of-frame message. Because this restriction is not enforced in asynchronous mode, the IG should 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 should go into a “ready” or “standby” state and begin sending start-of-frame messages to the Host. The *IG Mode* parameter within each **Start of Frame** packet (see Section 4.2.1) should be set to Reset/Standby to indicate this operational mode. Upon sending its first start-of-frame message, the IG should be considered mission-ready.

The IG will remain in Reset/Standby mode until it receives an **IG Control** packet (see Section 4.1.1) from the Host in which the *IG Mode* parameter is set to Operate. The Host should then wait for the IG to set a **Start of Frame** packet’s *IG Mode* parameter set to Operate before sending packets of any other type. The Host should continue to send **IG Control** packets while the IG is in Reset/Standby mode; however, any other data sent during this time will be ignored by the IG.

When the IG sends a **Start of Frame** packet in which the *IG Mode* parameter is set to Operate, the Host can 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 should then revert to its initial mission-ready condition. All entity instances should be removed from the scene; all in-process mission function requests should be purged; and all views, non-entity components, and environmental and weather conditions should be reset to their default states.

The Offline Maintenance mode can 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 can indicate this as being the current IG state.

2.9 Control Abstraction

Because CIGI is by design a generic interface, not all conceivable functional controls can be given a unique data packet. The **Component Control** packet (see Section 4.1.4) is provided as a general-purpose packet that can 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.

2.10 Interface Extensions

Although the CIGI is a robust interface, there may be times when a developer wishes to define a unique data packet format for a specific purpose. For this reason, CIGI has been designed to be extensible. Data packet opcodes 201 through 255 have been reserved for user-defined data packets.

To promote interoperability, a Host or IG device must gracefully ignore any packets it does not recognize. Further, although some device-specific functionality may be lost, the device must be able to revert back to the basic set of CIGI packets defined in this document.

Refer to Section 4.3 of this document for further details on user-defined data packets.

2.11 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 4.1.1) and **Start of Frame** (Section 4.2.1) packets. When a device receives a CIGI message, it can inspect the major version number to ensure that both devices are compatible. Some devices may be capable of automatically reverting to an older version of CIGI if necessary.

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 small changes may have been made to the interface but that the behavior defined in all prior minor versions has been preserved. A device whose counterpart uses an earlier minor version can look at the *Minor Version* parameter to determine whether some features and behaviors are not supported.

A major design goal for CIGI 3 is to build in provisions for 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 versions starting at Version 3:

1. Bit fields begin at the least significant bit available within the byte.
2. All bits marked “Reserved” must be set to zero (0) by the sender.
3. If a device receives a packet whose packet ID is not recognized, the packet should be ignored.
4. Any parameters added to a packet in future generations of CIGI will 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 will be ignored.

5. If a parameter's size must be increased, it will utilize any adjacent Reserved space or be moved the end of the packet as a new parameter. In the latter case, the sender may need to populate the original parameter depending upon the major and minor version numbers received from the other device.

6. The unit of measure for a parameter will not change. If a change of unit is required, a new parameter will be appended to the end of the packet. The sender should still populate the original parameter.

7. If a packet becomes obsolete, that packet's opcode will not be reused.

These guidelines are not intended to impose limitations that would be detrimental to the maturation of the interface. Although every attempt will 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 do not adhere to the above guidelines.

3. BASIC DEFINITIONS AND CONCEPTS

3.1 Entities

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

1. The entity has a distinct dynamic coordinate system (DCS).
2. The entity has a separate and unique tree in the IG scene graph. The tree may or may not include geometry.
3. The entity's tree can be transformed (i.e., translated, rotated, and scaled) independently from the rest of the scene graph.
4. The entity's tree can be moved within the scene graph's hierarchy. In other words, it can be attached to (become a branch of) and later detached from another entity's tree.

Entities can be used 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 tree 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 4.1.13 and 4.1.15, 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 trees or coordinate systems.

Views and view groups do possess distinct coordinate systems, but they do not have trees *per se*. View groups are hierarchical collections of views; however, no scene graph nodes exist.

Articulated parts cannot be detached from their respective super-trees and are therefore not entities. Moving parts on a model may be implemented as separate models rather than articulable submodels, however. In these instances, each part would be a child entity of the object containing the part.

See Section 4.1.2 for information on how the Host can instantiate and control entities on the IG with the **Entity Control** packet.

3.1.1 Ownship

The *Ownship* is a unique entity that represents the object being simulated by a trainer crew station, gaming platform, or other such device. Views attached to the Ownship correspond to the eyepoint of the pilot, driver, or other observer.

The Ownship is an entity, and as such, can be controlled with the **Entity Control** packet (Section 4.1.2).

3.1.2 Animations

An *animation* is an entity 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.

Depending upon the nature of the animation, the Host may modify certain characteristics by sending one or more **Component Control** packets (Section 4.1.4). 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 dynamic texture 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 can 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 (e.g., 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 tree 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 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.

Note that some animated components may be implemented as submodels of an entity model rather than as separate entities. 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 4.1.4).

3.2 Views

A *view* is a projection of a three-dimensional volume onto a two-dimensional viewing plane. CIGI 3 supports perspective and orthographic parallel views. Oblique parallel views are not directly supported, but may be implemented by using the **Component Control** packet (Section 4.1.4) to specify the direction of projection relative to the projection plane.

3.2.1 Viewing Volumes

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

3.2.1.1 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.

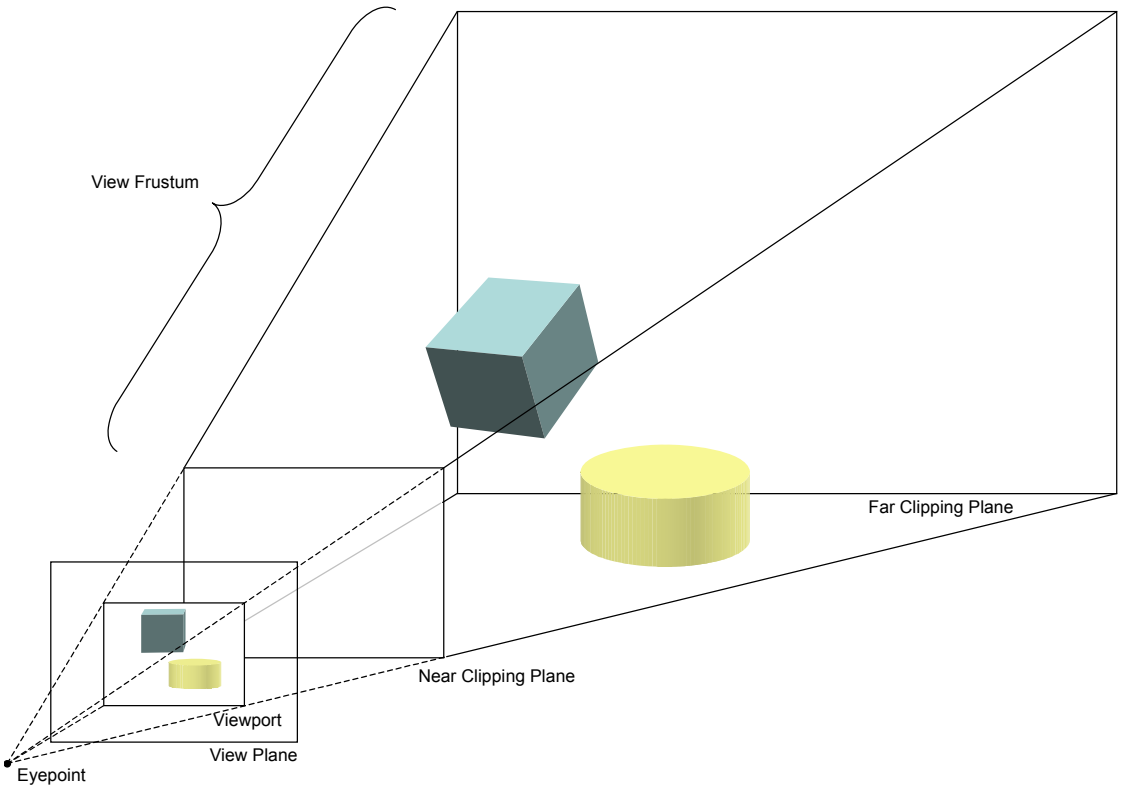


Figure 12 – Perspective Projection onto a Viewport

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 can be achieved.

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.

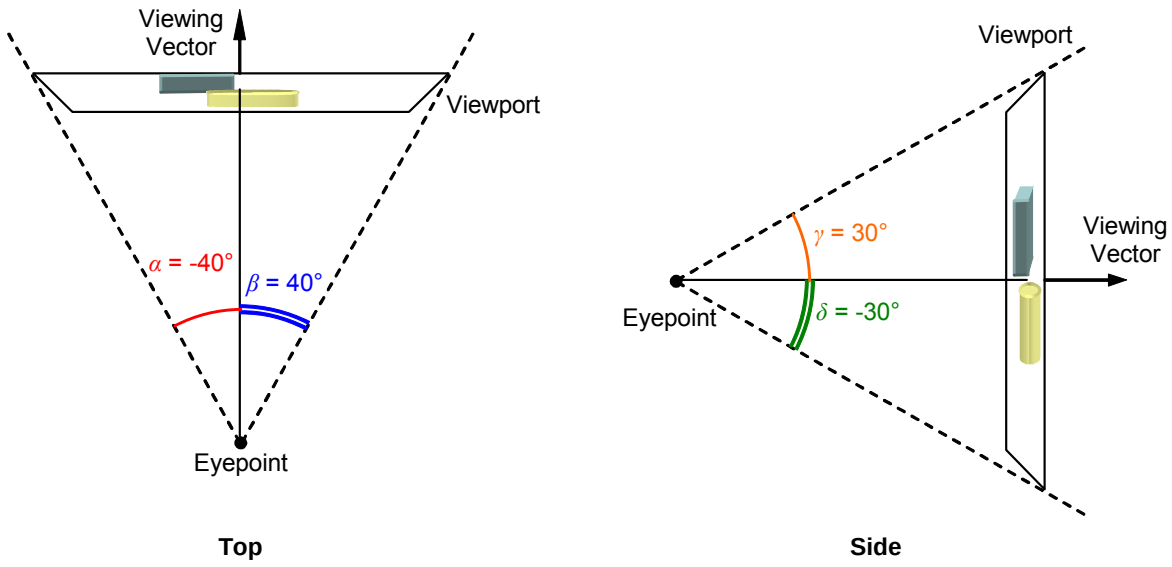


Figure 13 – View Definition Half-Angles

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:

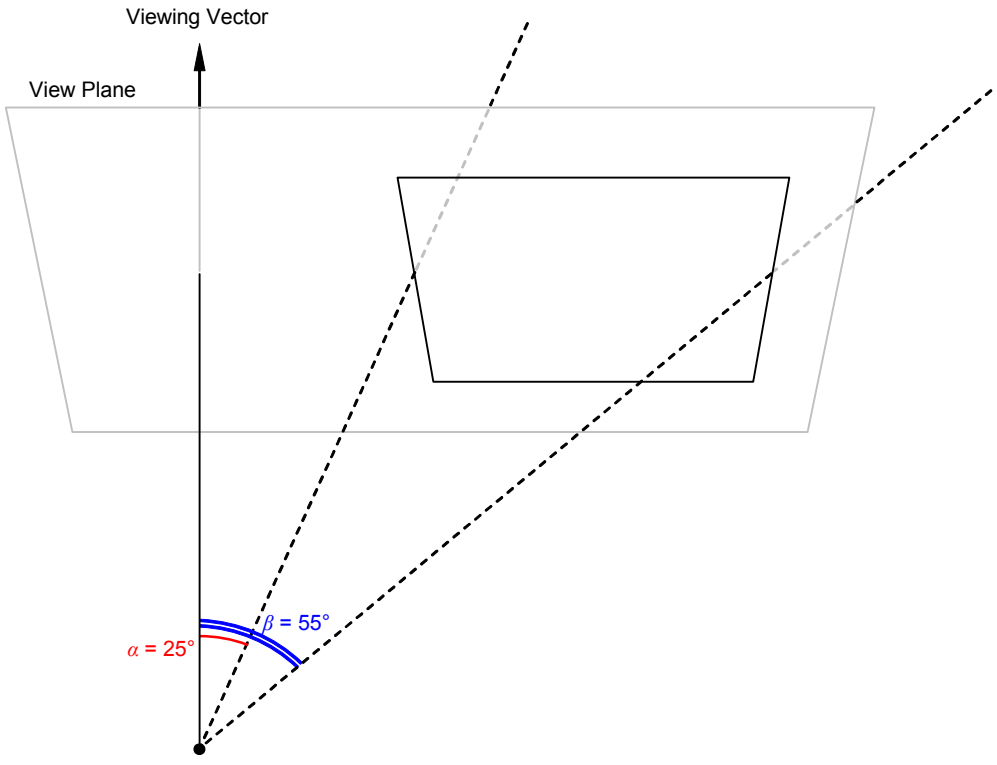


Figure 14 – Example of an Oblique Perspective View

The size of the view frustum can be set via the **View Definition** packet. Refer to Section 4.1.21 for details about this packet.

3.2.1.2 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.

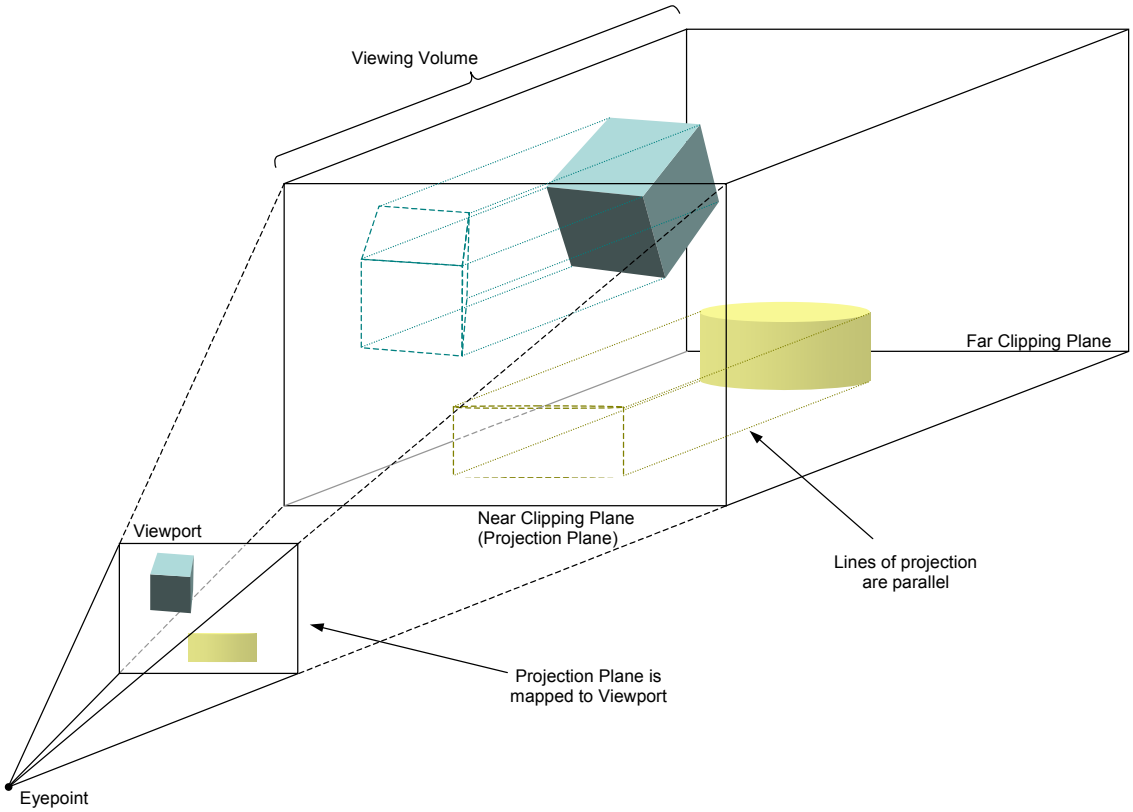


Figure 15 – Orthographic Parallel Projection onto a Viewport

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.

3.2.2 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 can be performed upon them simultaneously.

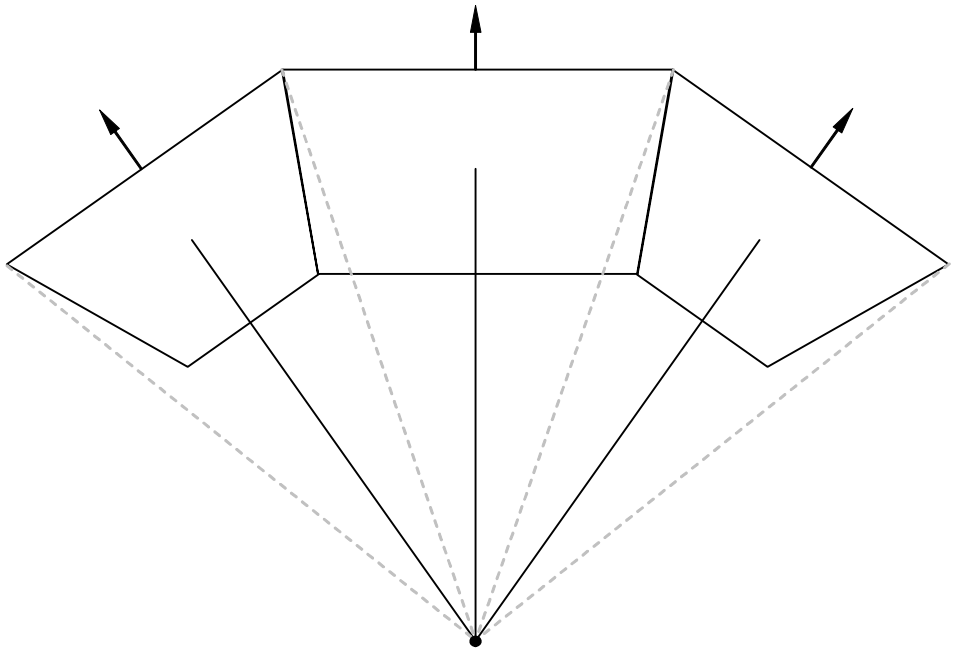


Figure 16 – Example of a View Group

The latter technique has several advantages. Perhaps most importantly, it reduces the network load because a single **View Control** packet (see Section 4.1.13) can 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 can be assigned to a view group by setting the *Group ID* parameter of the **View Definition** packet (see Section 4.1.21). Once the group assignment has been made, the view may be controlled with the **View Control** packet (see Section 4.1.16) either individually or as part of the group. Refer to the indicated sections for more information on these packets.

3.3 Symbols

A *symbol* as defined in this document is a single drawing primitive or a group of drawing primitives that can be drawn on a symbol surface within a particular view.

Symbols can be defined by the Host using one of the symbol definition packets as described in Sections 4.1.30 through 4.1.32. Each symbol is identified by a unique symbol ID. If the Host uses two visually identical symbols at the same time, both symbols must be defined with separate identifiers.

Each instance of a symbol must be defined independently. If the Host uses two visually identical symbols at the same time, both symbols must 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.

3.3.1 Symbol Surfaces

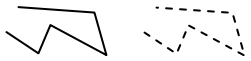
A *symbol surface*, or simply *surface*, is a rectangular, two-dimensional drawing region on a virtual plane on which symbols may be drawn. A surface is placed in 3D space relative to a particular entity or coincident with the near clipping plane of a particular view. The extents of the rectangular drawing region are defined with respect to one of the coordinate systems described in Sections 3.4.4.1 through 3.4.4.3.

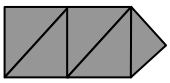
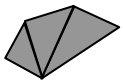
3.3.2 Symbol Primitives

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).	Hello World!
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.	

Drawing Primitive	Description	Example
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.
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 can include only one type of primitive. Compound symbols comprised of multiple primitive types must 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 is defined, no primitives can be added, removed, or modified without destroying and redefining the symbol.

3.3.3 Symbol Templates

The IG may create one or more templates defining symbol geometry. These symbol templates can be instantiated with a **Symbol Clone** packet (4.1.33). Once instantiated, the new symbols are treated like any other symbol and can be manipulated through **Symbol Control** or **Short Symbol Control** packets.

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

3.4 Coordinate Systems

3.4.1 Geodetic Coordinate System

CIGI 3 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 3.4.1.1 and 3.4.1.2 describe geodetic position and orientation, respectively.

The default Earth Reference Model (ERM) for CIGI 3 is WGS 84. The Host can define a new reference ellipsoid by sending an **Earth Reference Model Definition** packet (Section 4.1.19) to the IG.

3.4.1.1 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.

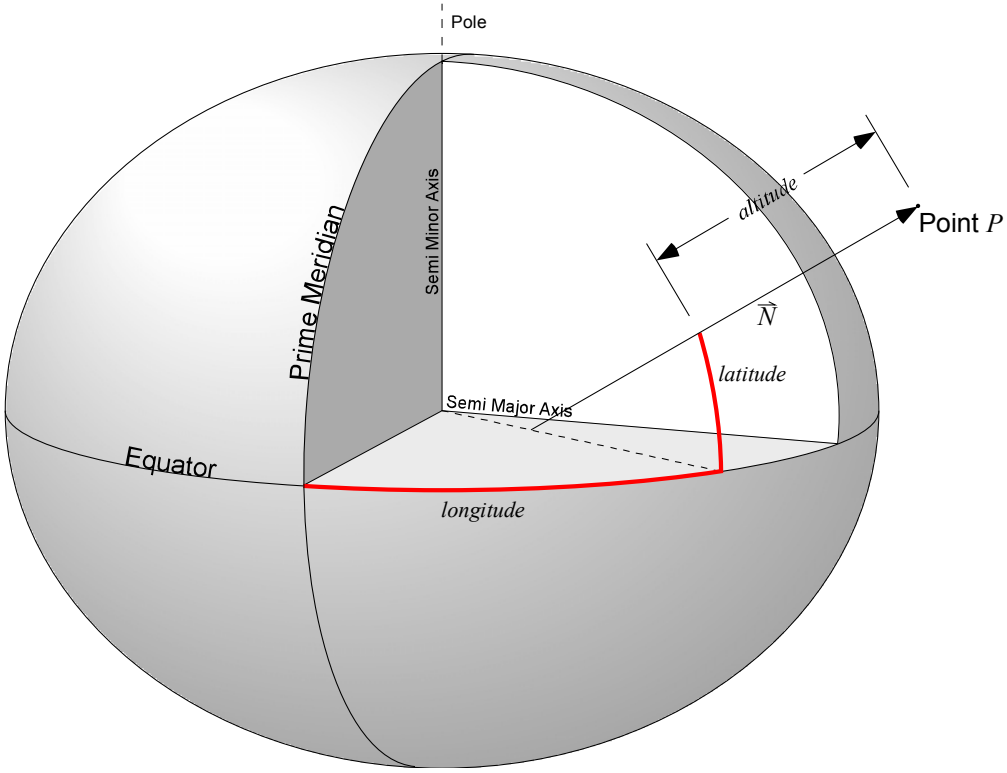


Figure 17 – Position within Geodetic Coordinate System

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 3.4.1.2.

3.4.1.2 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. A right-hand coordinate system can be 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 and is shown in Figure 18.

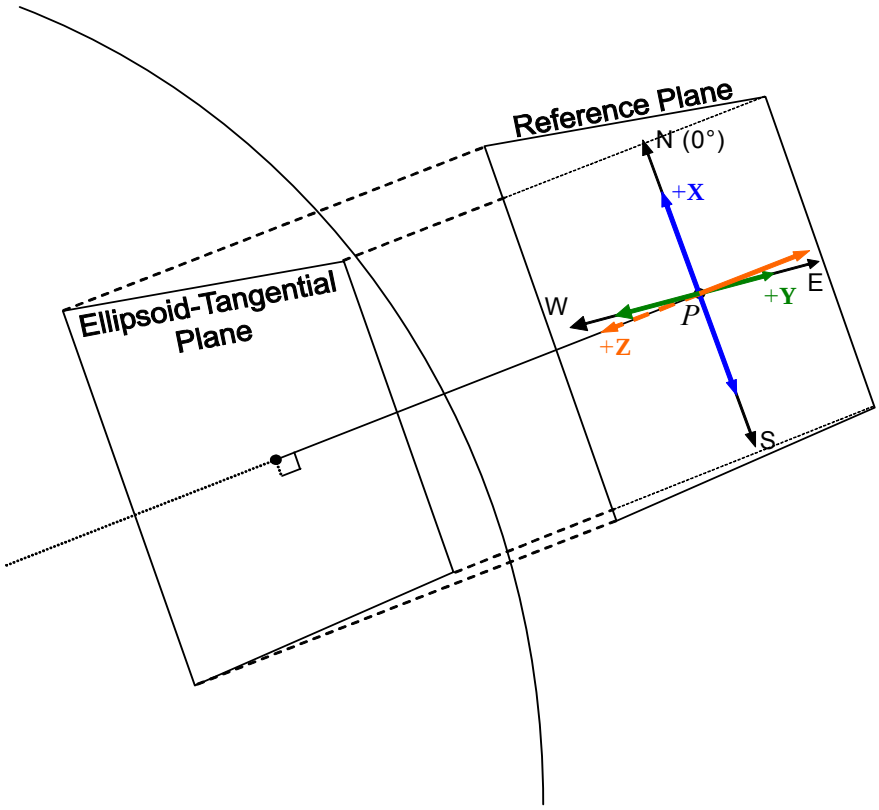


Figure 18 – Local Geodetic Reference Plane with NED Coordinate System

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

An entity's yaw, ψ, is the measure of the angle formed from True North to the entity's +**X** axis. This angle is specified in degrees and is positive clockwise if looking along the +**Z** axis.

An entity's pitch, θ, is the measure of the angle between the reference plane and the entity's +**X** axis. This angle is specified in degrees and is positive above (away from the ellipsoid) the reference plane.

Roll, φ, is the measure of the angle between the reference plane and the entity's +**Y** axis along a plane perpendicular to the entity's **X** axis. In other words, it is the angle of rotation about the **X** axis *after yaw and pitch have been applied*. Roll is also specified in degrees and is positive clockwise from the point of view of looking along the +**X** axis.

Figure 19 illustrates the rotation about each axis:

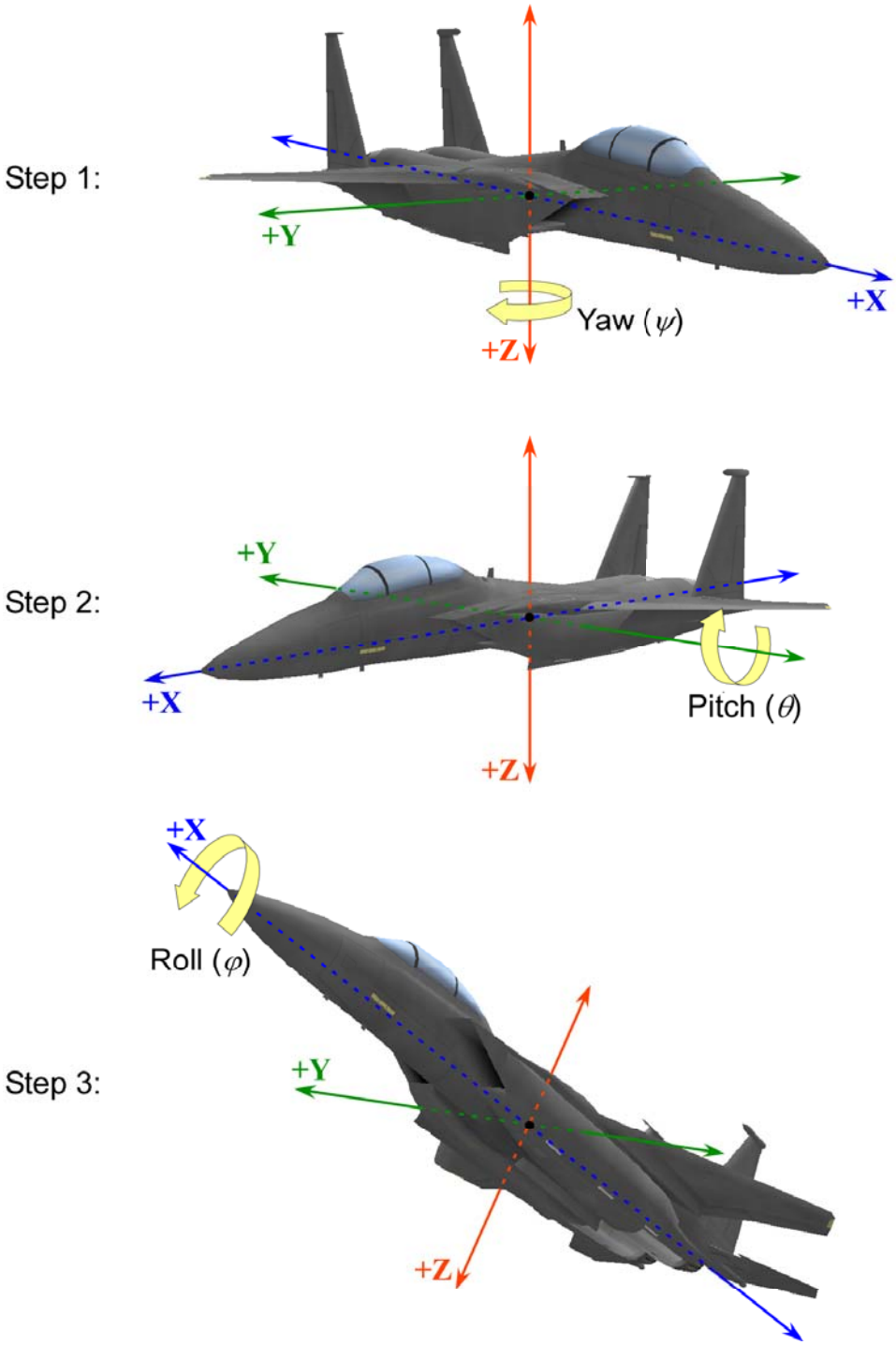


Figure 19 – Rotation in NED Coordinate System

3.4.2 Entity Coordinate System

Each entity has a local NED 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 extends out the "front" of the entity (e.g., the nose of an aircraft). The +**Y** axis extends out the right side, and the +**Z** axis extends out the bottom of the entity.

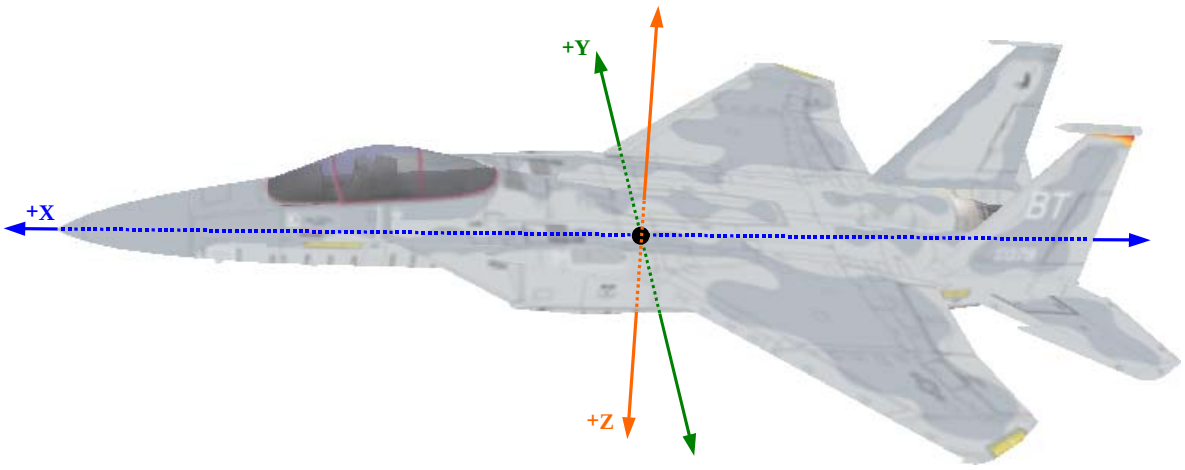


Figure 20 – Local Entity Coordinate System

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

3.4.2.1 Position

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

3.4.2.2 Orientation

Rotation with respect to an entity's coordinate system is 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 as shown in Figure 19.

3.4.3 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 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.

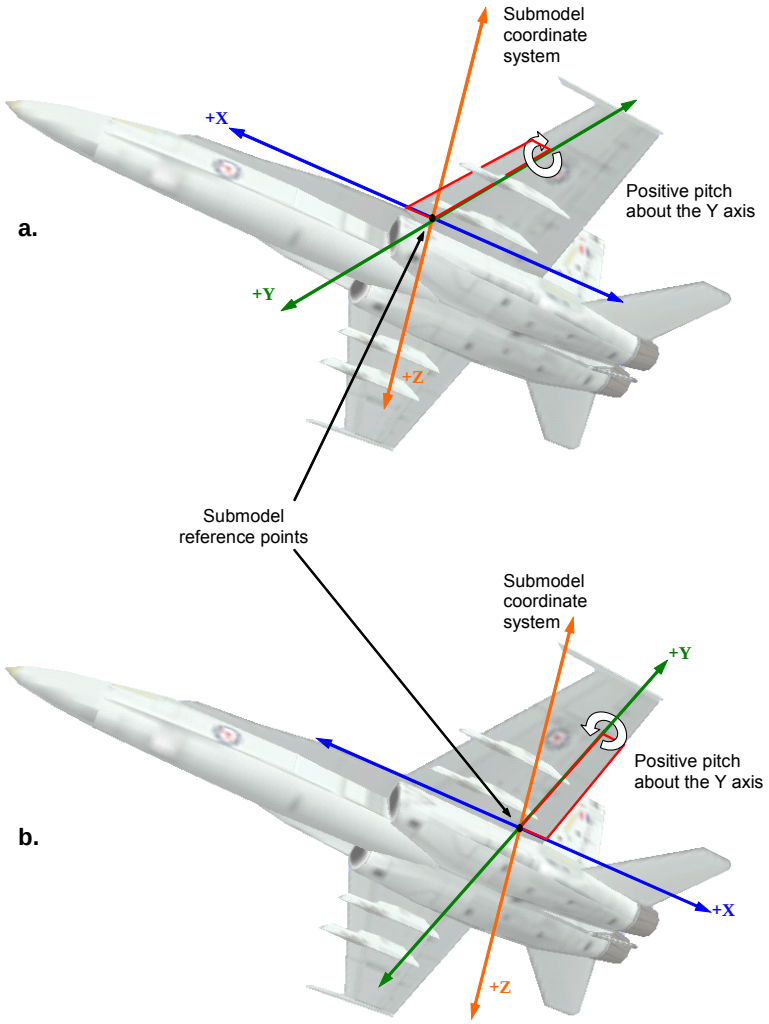


Figure 21 – Examples of Submodel Coordinate Systems

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

Rotations applied to a submodel are not cumulative. In other words, specifying a rotation and translation will override any previous values.

Section 4.1.6 describes the use of the **Articulated Part Control** packet in manipulating submodels.

3.4.4 Symbol Surface Placement Coordinate Systems

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

3.4.4.1 Entity Coordinate System

A symbol surface that is attached to an entity and is not a billboard uses the entity's local coordinate system to specify position and orientation relative to that entity. Thus, position and rotation of such a symbol is specified as described in Section 3.4.2.

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

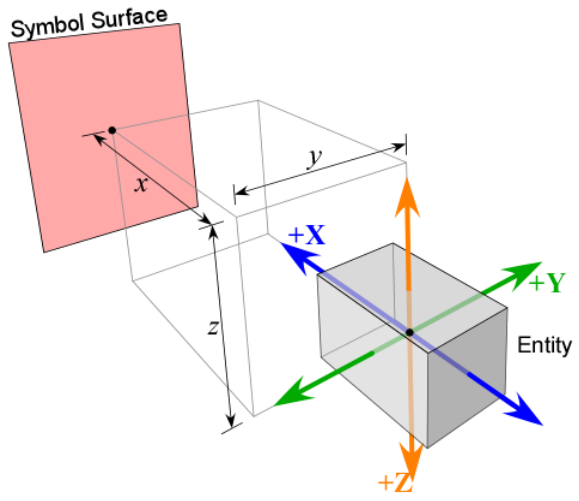


Figure 22 – Entity Coordinate System for Placing Symbol Surfaces

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:

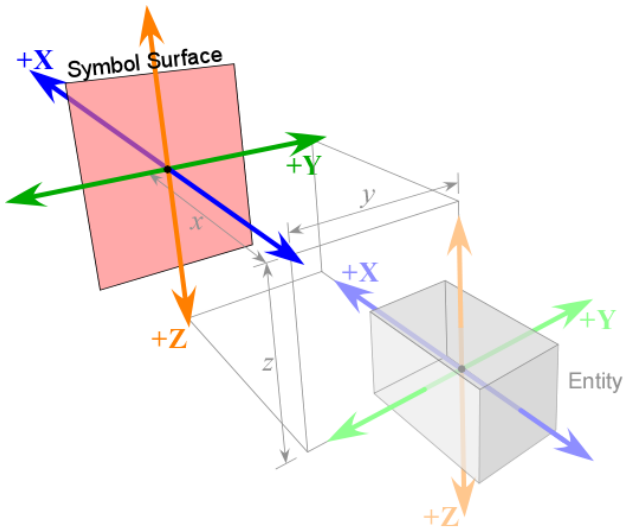


Figure 23 – Entity-Attached, Non-Billboard Symbol Surface Reference Coordinate System

This reference coordinate system is used to specify the size and rotation of the symbol surface. The surface's width is measured along the reference **Y** axis, and its height is measured along the reference **Z** axis. Yaw, pitch, and roll rotations are about the **Z**, **Y**, and **X** axes, respectively and in that order, as shown in Figure 24:

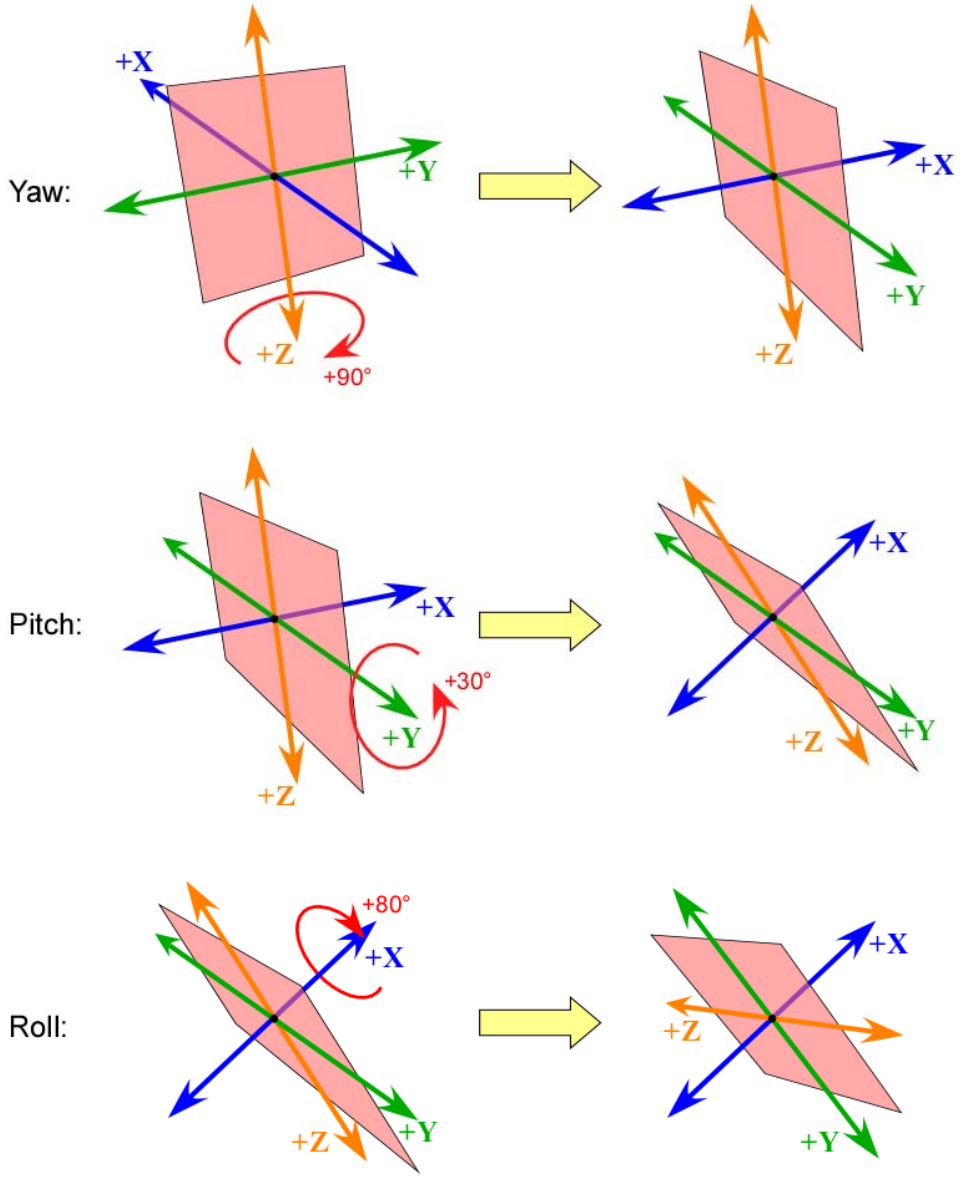


Figure 24 – Rotation of an Entity-Attached Symbol Surface

First, in the above example, 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 3.4.2.2.

3.4.4.2 Symbol Surface Billboard Coordinate System

A symbol surface that is attached to an entity and is a billboard is always on a plane parallel to the view's viewport. A normal vector emanating from the center of the symbol surface toward the eyepoint is parallel to, but

in the opposite direction of, the view's viewing vector. Thus, the surface's **U** axis (see Section 3.4.5.1) 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 is 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 cannot 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 extends along the aforementioned normal vector and is positive in the direction toward the eyepoint. The **X** axis is horizontal relative to the view and is positive to the right from the perspective of the eyepoint. Likewise, the **Y** axis is vertical relative to the view and is 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.

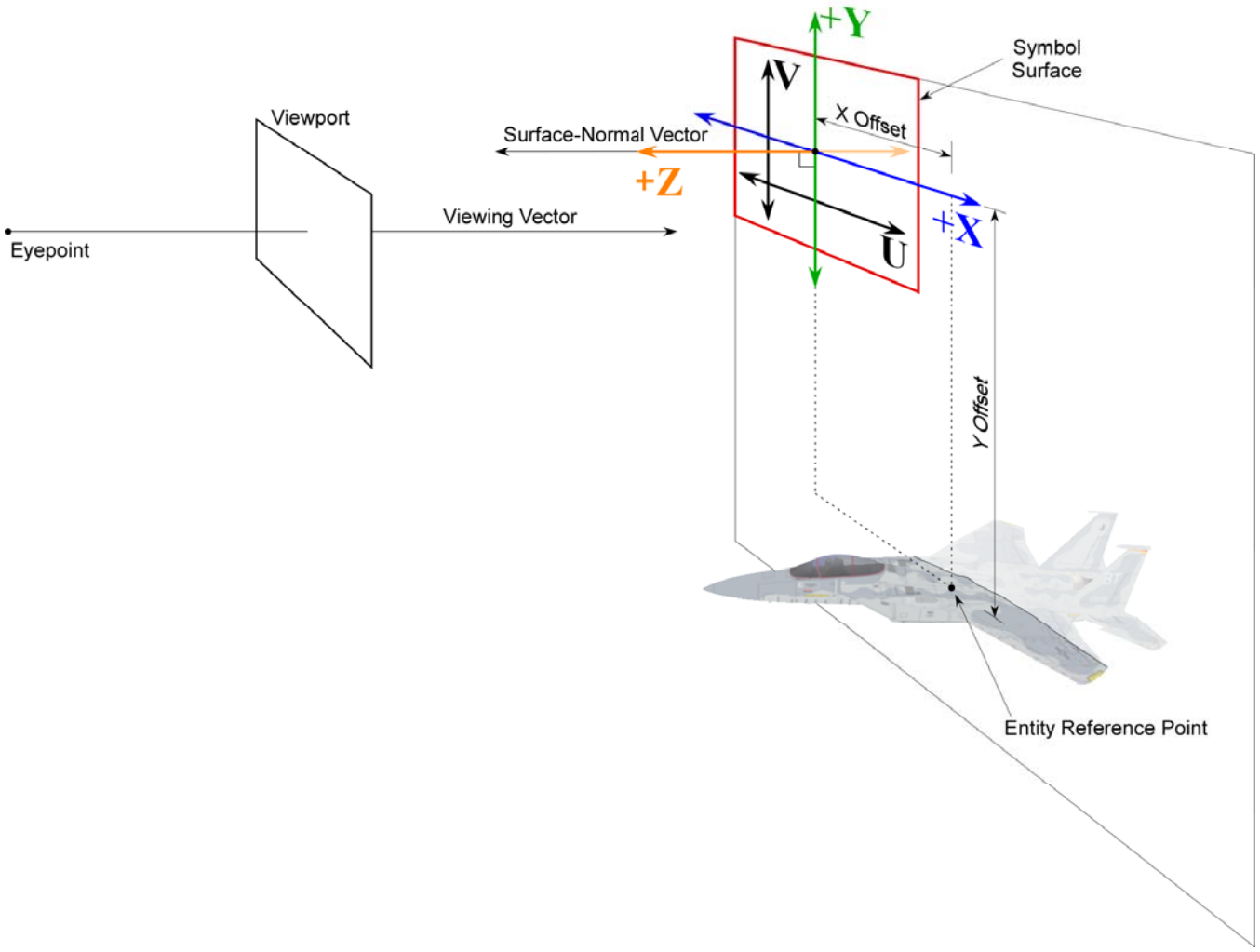


Figure 25 – Symbol Surface Billboard Coordinate System

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. 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.

3.4.4.3 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 are defined in terms of the width and height of the viewport. The coordinates (0, 0) and (1.0, 1.0) are mapped to the lower-left and upper-right corners of the viewport, respectively. The symbol surface is placed with respect to the 2D Cartesian coordinate system implied by those points as shown in Figure 26:

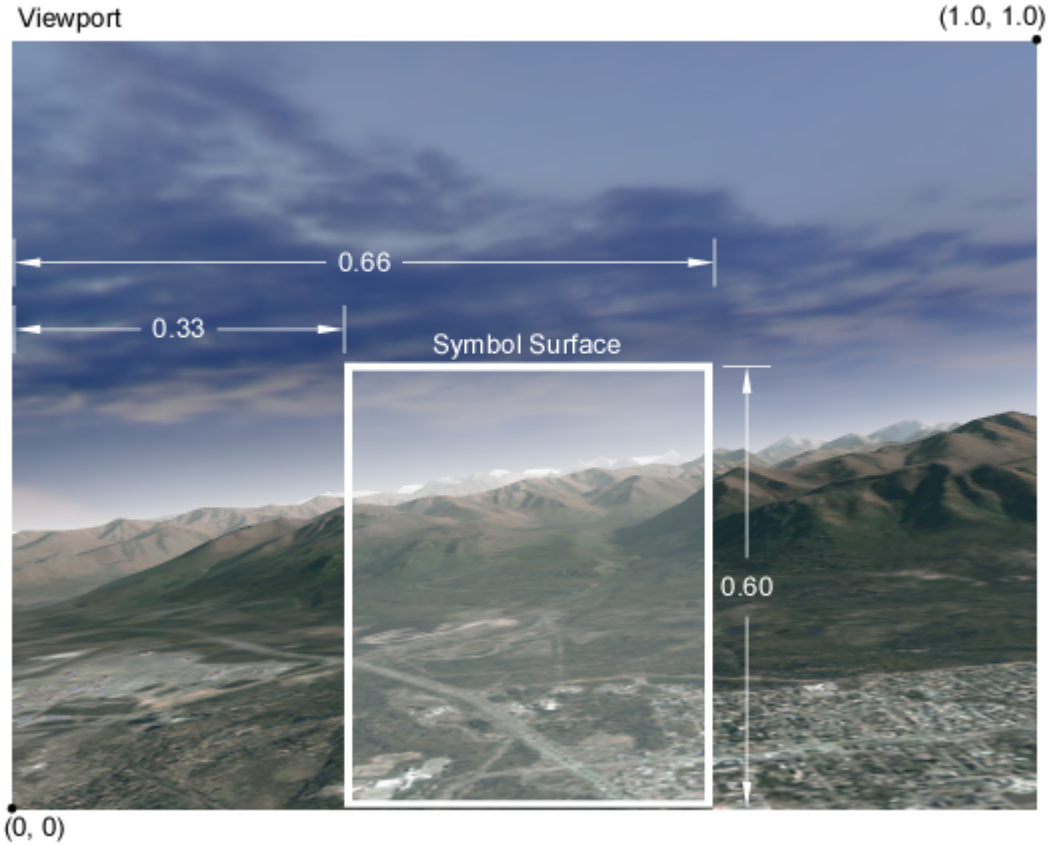


Figure 26 – Example of a View-Attached Symbol Surface

In the example above, the symbol surface is defined with left and right sides at 0.33 and 0.66 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 0.60 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.

3.4.5 Symbol Placement Coordinate Systems

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

3.4.5.1 Symbol Surface 2D Coordinate System

A symbol surface is associated with a 2D Cartesian (UV) coordinate system 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 will correspond to the left, right, top, and bottom boundaries of the surface. A symbol surface with a UV coordinate system is depicted in Figure 27.

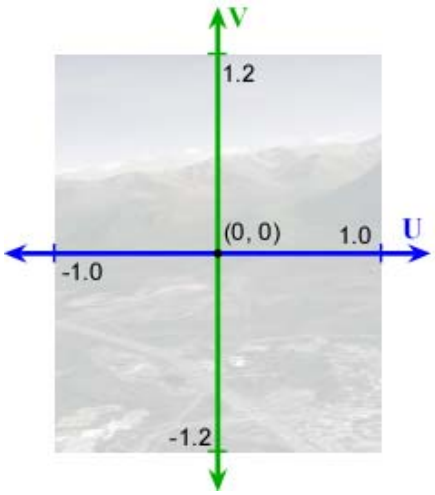


Figure 27 – Example of a Symbol Surface 2D Coordinate System

In the example above, the symbol surface's horizontal and vertical extents have been chosen to be ±1.0 and ±1.2 units, respectively, to match the aspect ratio of the surface. This produces units that are "square" and places the origin 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.

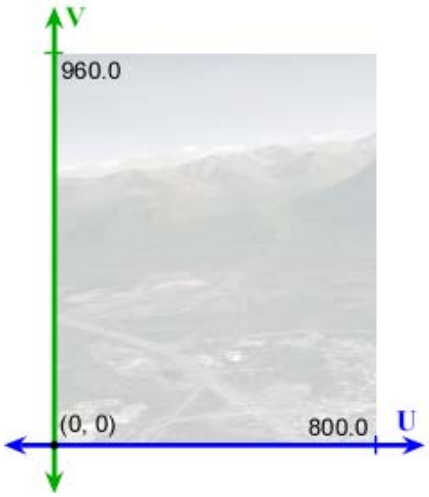


Figure 28 – Example #2 of a Symbol Surface 2D Coordinate System

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 must be defined to be less than the right side and that the bottom must be defined to be less than the top.

A symbol's primitives' dimensions are 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 will extend 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 will be drawn as an ellipse.

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

The position of a symbol's local 2D coordinate system (see Section 3.4.5.2) is specified in this UV coordinate system, as well. The symbol's reference point is placed at the origin of this local coordinate system, and the symbol is rotated about this reference point as described below.

Note that scaling operations are performed after the reference coordinate system is translated and rotated.

3.4.5.2 Symbol 2D Coordinate System

Every symbol has a local Cartesian coordinate system. This coordinate system is used for rotating the symbol and for positioning the symbol's primitives and child symbols. The origin of this coordinate system always corresponds 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:

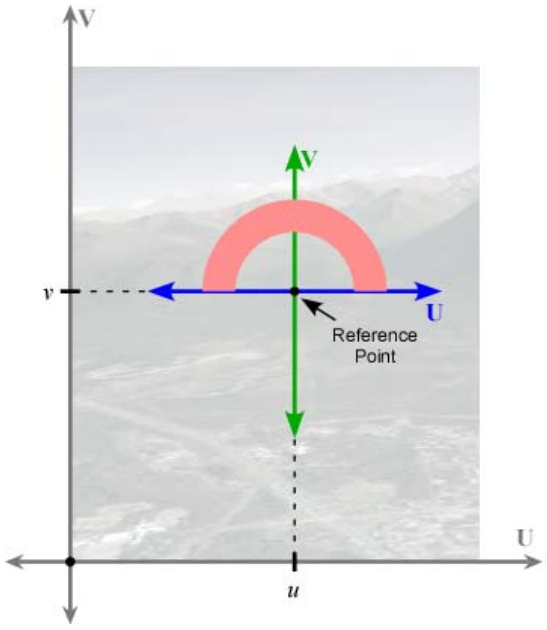


Figure 29 – Symbol 2D Coordinate System Relative to a Symbol Surface

The origin of the top-level symbol's local coordinate system, as well as the symbol's reference point, is 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. The horizontal and vertical units for both sets of axes are the equal in length.

Two or more symbols may be arranged in a hierarchy. A child symbol's local coordinate system is 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.

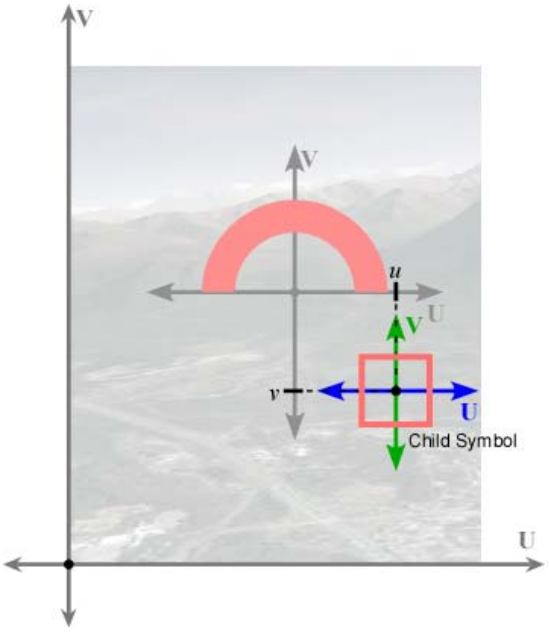


Figure 30 – Child Symbol Coordinate System Relative to Parent

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

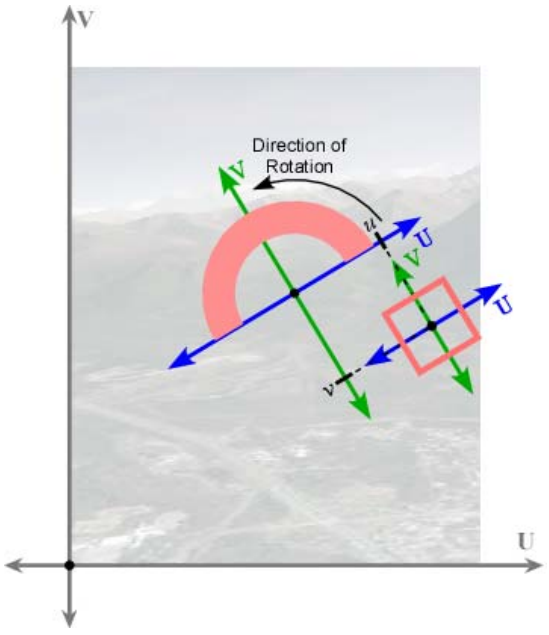


Figure 31 – Rotated Parent Symbol with Child

Scaling is 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.

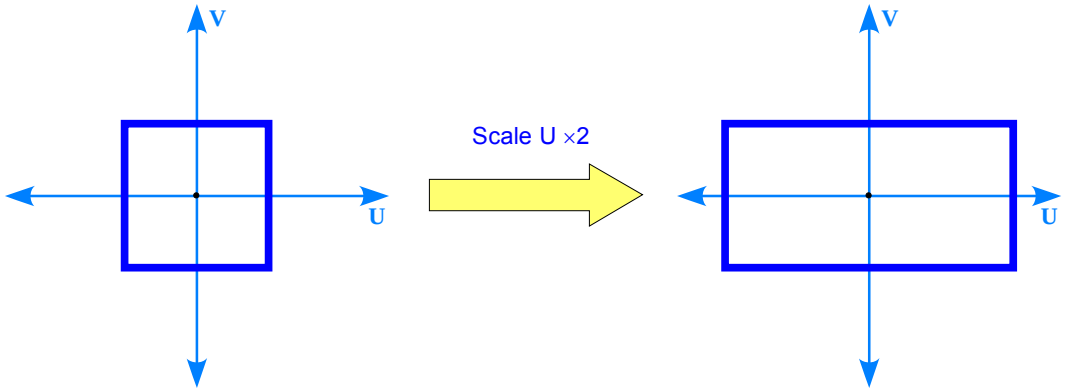


Figure 32 – Scaling a Symbol

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

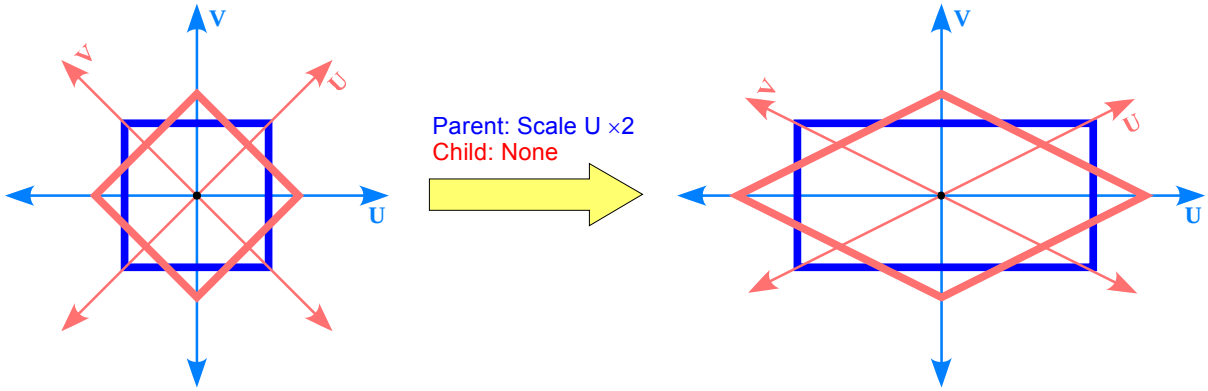


Figure 33 – Scaling a Parent Symbol

Figure 34 shows the result of scaling only the child.

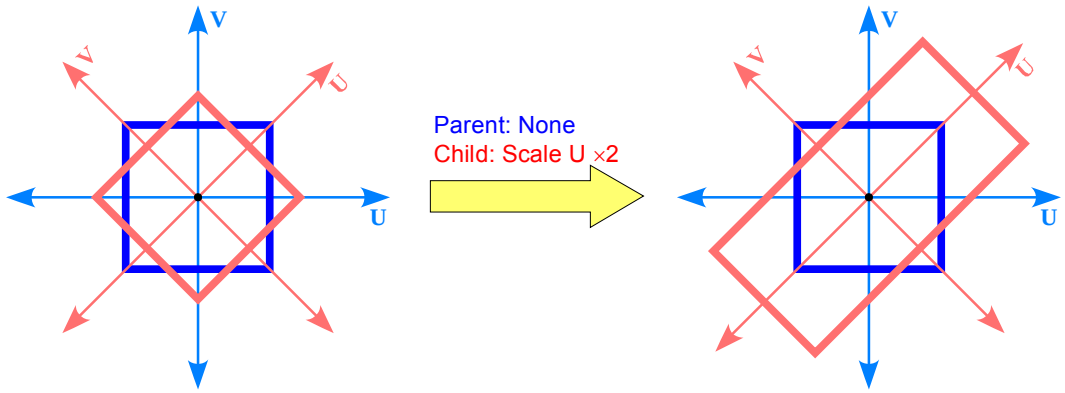


Figure 34 – Scaling a Child Symbol

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

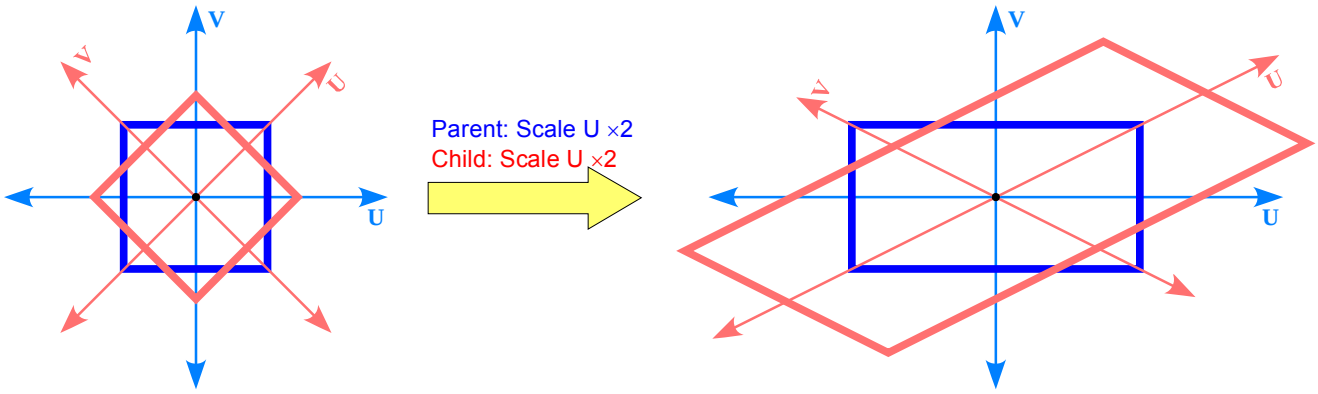


Figure 35 – Scaling a Parent Symbol and a Child Symbol

Note that scaling factors applied to any given local coordinate system are not 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 is discarded. The appearance of the symbol will still be affected by the parent's scale, however.

4. DATA PACKET REFERENCE

This portion of the ICD describes each packet in the Common Image Generator Interface. Section 4.1 describes the CIGI packets used in Host-to-IG messages. Section 4.1.29 describes the packets used in IG-to-Host messages. Section 4.3 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.

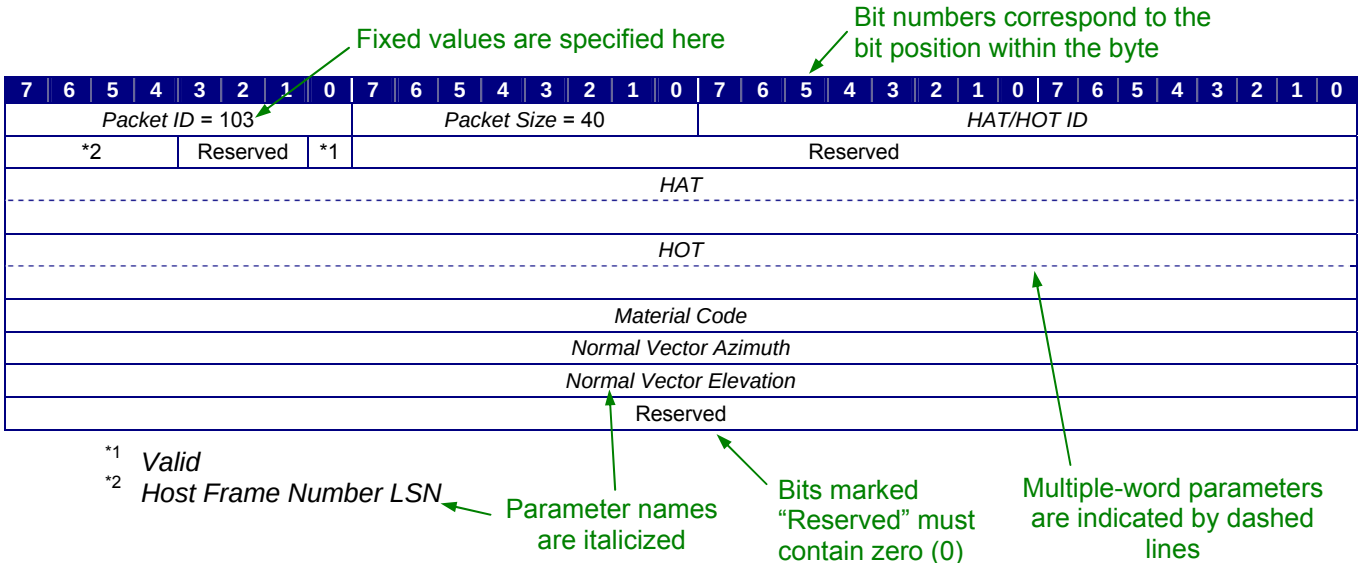


Figure 36 – Example of Packet Structure Diagram

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 will not fit within the space allotted in the table are noted below the diagram. Bits marked "Reserved" are allocated for future use and must 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 ID	**Low Address**	*Packet ID*
Packet Size		*Packet Size*
HAT/HOT ID (LSB)		*HAT/HOT ID* (MSB)
HAT/HOT ID (MSB)		*HAT/HOT ID* (LSB)
Valid/Frame Number LSN		*Valid/Frame Number LSN*
0		0
0		0
0		0
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 9, 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 ID Type: unsigned int8 Units: N/A Value: 103	This parameter specifies the packet's opcode, which uniquely identifies the packet. In this example, the value 103 indicates that this is a **HAT/HOT Extended Response** packet. This is a dimensionless value.
Packet Size Type: unsigned int8 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.
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 2.7. 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.

4.1 Host-to-IG Packets

4.1.1 IG Control

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

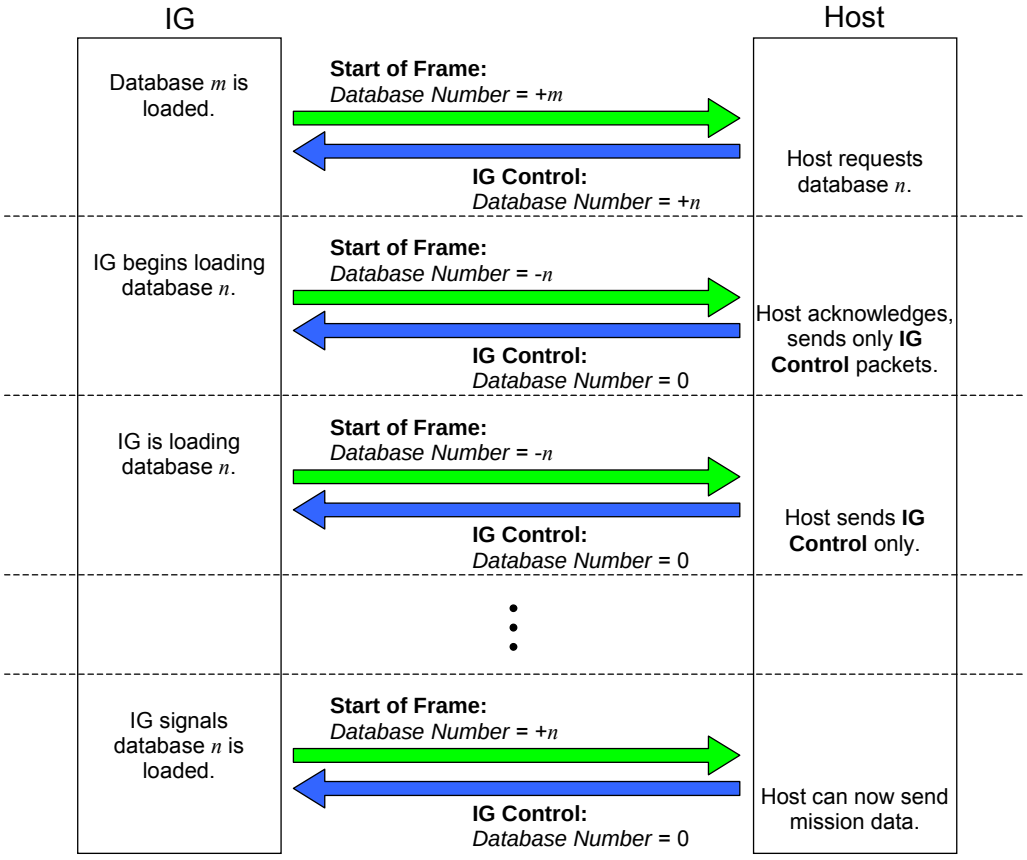


Figure 38 – Database Loading Sequence in Synchronous Mode

The IG will 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 will 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 should ideally send only **IG Control** packets at this time.

After the IG receives the acknowledgement, it will 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 is now considered mission-ready and can receive mission data from the Host.

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

When using a global database, the IG will set the *Database Number* of the **Start of Frame** packet to zero (0). When the Host detects a zero in this parameter, it should 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:

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 ID = 1								*Packet Size* = 24								*Major Version* = 3								*Database Number*							
Minor Version				*3	*2	*1		Reserved								*Byte Swap Magic Number*															
Host Frame Number																															
Timestamp																															
Last IG Frame Number																															
Reserved																															

*1 *IG Mode*
*2 *Timestamp Valid*
*3 *Extrapolation/Interpolation 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 ID Type: unsigned int8 Units: N/A Value: 1	This parameter identifies this data packet as the **IG Control** packet. The value of this parameter must be 1.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
Major Version Type: unsigned int8 Units: N/A Value: 3	This parameter indicates the major version of the CIGI interface that is currently being used by the Host. The IG can use this number to determine concurrency. The Host must set the value of this parameter to 3.

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. Setting this parameter to a non-zero value will cause the IG to begin loading the database that corresponds to that value. If the number corresponds to the current database, the database will be reloaded. The IG will 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 *Database Number* parameter of the **IG Control** packet to zero (0) to prevent continuous reloading of the database on the IG. The IG will ignore this parameter while in Reset/Standby mode. Refer to Section 4.2.1 for more information on the **Start of Frame** packet.
IG Mode Type: unsigned 2-bit field Units: N/A Values: 0 Reset/Standby 1 Operate 2 Debug Default: 0	This parameter dictates the IG's operational mode. The Host can initiate a mode change by setting this parameter to the desired mode. When the IG completes the mode change, it will set 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 44.
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. Because the *Timestamp* parameter is required for asynchronous operation, *Timestamp Valid* must be set to Valid (1) in this mode.
Extrapolation/Interpolation 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 are enabled. If this parameter is set to Disable (0), then extrapolation or interpolation is disabled for all entities. If this parameter is set to Enable (1), then extrapolation or interpolation is determined on a per-entity basis by the *Linear Extrapolation/Interpolation Enable* flag in the **Entity Control** packet.

Parameter	Description
Minor Version Type: 4-bit field Units: N/A Value: 2	This parameter indicates the minor version of the CIGI interface that is currently being used by the Host. The IG can use this number to determine concurrency.
Byte Swap Magic Number Type: unsigned int16 Units: N/A Values: 8000h No byte swap (see note at right) 80h Byte swap	This parameter is used by the IG to determine whether it needs to byte-swap incoming data. Refer to Section 2.5 for details on this mechanism. Note: The Host *must* set this value to 8000h, or 32768.
Host Frame Number Type: unsigned int32 Units: N/A	This parameter uniquely identifies a data frame on the Host. The Host should 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.
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. This will enable the IG to correct for latencies as described in Section 2.2.1. 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 use of this parameter is required for asynchronous operation. The use of this parameter is optional for synchronous operation. If this parameter does not contain a valid timestamp, the *Timestamp Valid* parameter should be set to zero (0).
Last IG Frame Number Type: unsigned int32 Units: N/A	This parameter contains 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.

4.1.2 Entity Control

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

Each entity is 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, the IG will create it.

When the IG creates an entity, it makes a copy of the geometry corresponding to the value of the *Entity Type* parameter. This copy exists as a unique and independent tree within the scene graph; therefore, any operations that modify an entity's tree (e.g., part articulations) affect only that entity and its children.

Figure 40 illustrates unique Entity IDs being assigned to multiple entities of the same type. The number assignments in this example are hypothetical.

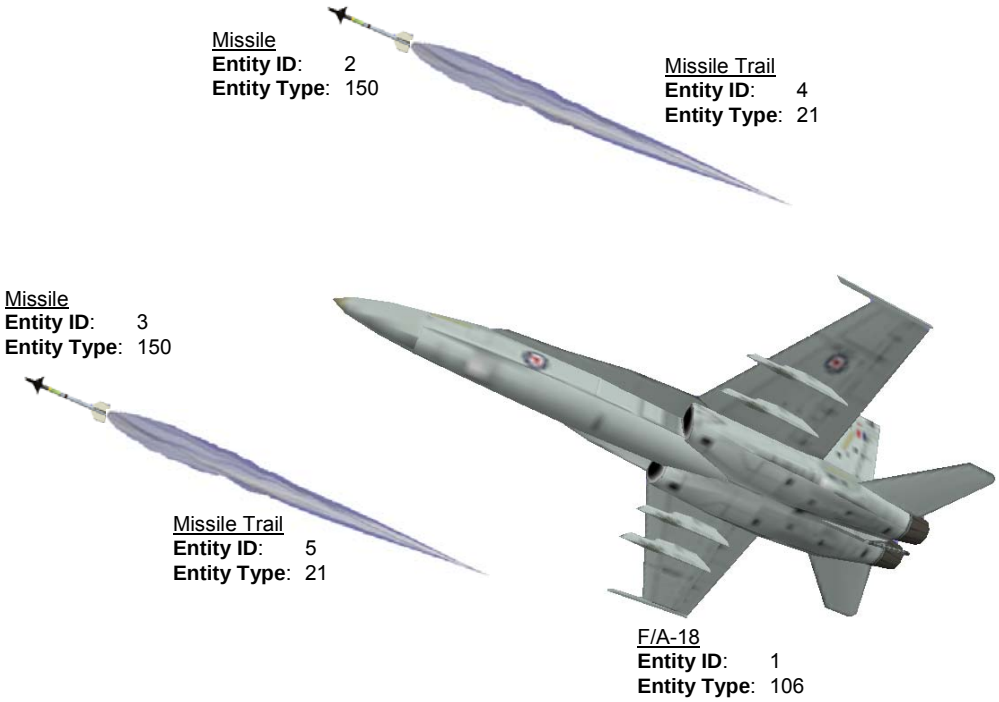


Figure 40 – Example of Entity Definitions

Entities can be attached to one another in a hierarchical relationship. In such a hierarchy, a child entity's position is specified relative to its parent's coordinate system. The Host needs only to control the parent entity in order to move all lower entities in the hierarchy as a group. No explicit manipulation of a child entity is necessary unless its position and attitude change with respect to its parent.

In the example above, 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 41 illustrates the parent-child hierarchy before and after a missile is launched:

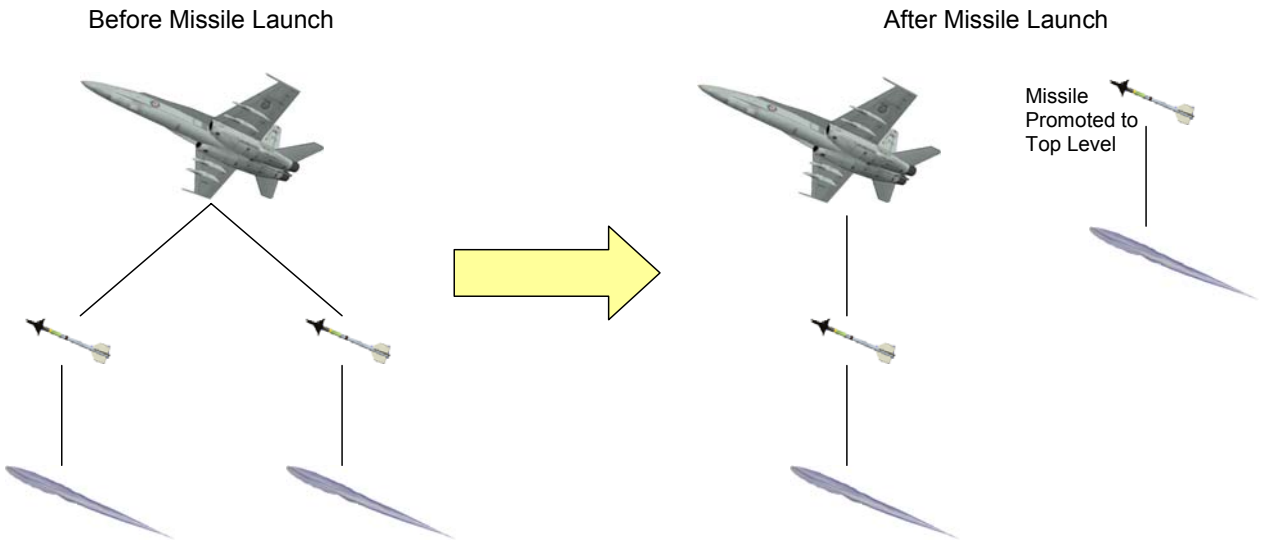


Figure 41 – Example of Child Entity Detachment

The *Attach State* parameter of the **Entity Control** packet determines whether an entity is attached to a parent. If this parameter is set to Attach (1), the entity is attached to the entity specified by the *Parent ID* parameter.

The *Entity State* field is used to control when an entity is visible and when its geometry is loaded and unloaded. When an entity is created, the *Entity State* field can be set to Active to specify that the entity should be added to the scene as soon as the model geometry is loaded. The entity and any children can be made invisible at any time by setting *Entity State* to Inactive/Standby. When the entity is no longer needed, *Entity State* can be set to Destroyed to direct the IG to unload the geometry and free any memory allocated for the entity. Any children attached to the entity are also destroyed.

Models can be preloaded to increase the speed at which they can be initially displayed. For example, when an aircraft fires a missile, a new entity would need to be created for that missile. Unless the missile geometry is cached, the IG must load the model from its hard disk. Because of its tremendous 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 can already exist in memory and be instantly activated within the scene graph when needed. To accomplish this, the *Entity State* flag could be set to Inactive/Standby when the missile is created. Later, when the missile is needed, an **Entity Control** packet for that entity would be sent containing the proper positional data and with the *Entity State* flag set to Active.

An entity can also be made invisible by setting the *Alpha* parameter to zero (0). This parameter specifies an alpha value to be applied to the entity's geometry. The *Inherit Alpha* parameter indicates whether a child entity's alpha value is 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 *Entity State* parameter enables or disables the entity geometry in the scene graph. The entity would not be included in line of sight and collision testing, nor would any transformations be applied. Any children would also be disabled. The *Alpha* parameter, on the other hand, merely affects the opacity of the specified entity.

The positions of top-level entities (i.e., those entities that are not children) are always specified as a geodetic latitude, longitude, and altitude (see Section 3.4.1.1). The positions of child entities are always specified with respect to the parents' NED body coordinate system (see Section 3.4.2).

In certain instances, it is desirable for the IG to "clamp" the entity to the surface of the terrain. This can be used as an alternative to using HOT requests and responses to determine ground elevation and slope below the entity. If the *Ground/Ocean Clamp* parameter is set to Non-Conformal (1) or Conformal (2), the *Altitude* parameter specifies an offset above the ground or sea surface height. This is 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* specify rotational offsets if ground or ocean clamping is enabled.

The *Animation State* parameter is used to control the playback state of animation entities. To start the animation sequence, the Host will send an **Entity Control** packet with its *Entity State* set to Active and its *Animation State* parameter to either Play or Resume. The Host may explicitly stop the animation at any time by setting the *Animation State* to Stop. Setting the parameter to Pause freezes the animation sequence at the current frame. Setting the parameter to Resume in a subsequent frame will resume the animation from its paused state; setting it to Play will play the animation again from its initial state. Setting *Animation State* to Play during playback will restart the animation.

Note that setting the *Animation State* parameter to Stop will 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) will stop producing new particles or segments; however, existing particles or segments will 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:

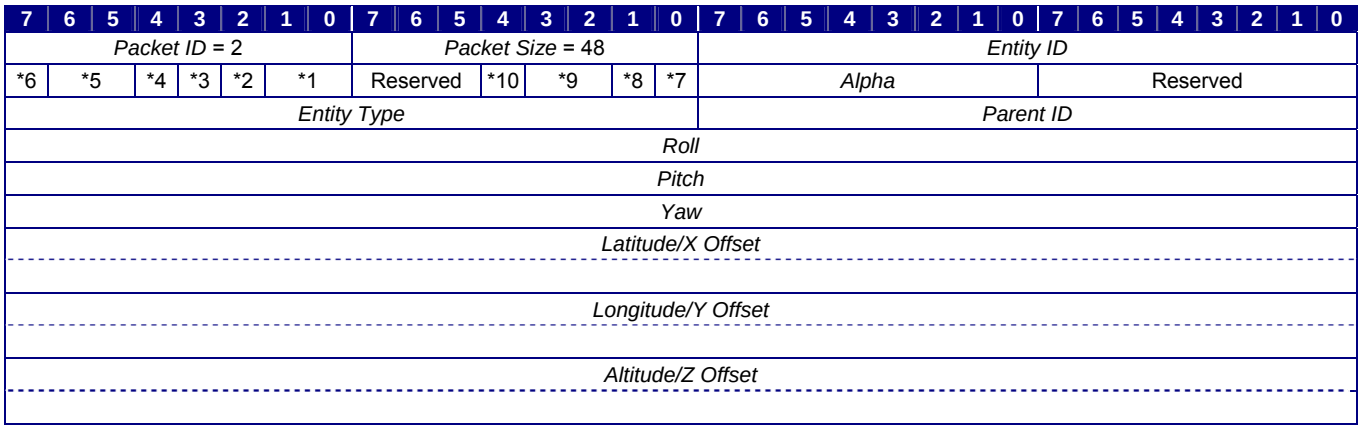
Table 6 – Animation State Summary

Current State	New *Animation State* Value	Effect
Stopped	Stop	None
	Pause	None
	Play	Plays from beginning
	Continue	Plays from beginning
Playing	Stop	Stops at current frame; Stops emitting particles/segments
	Pause	Pauses at current frame; Freezes all particles
	Play	Restarts from beginning
	Continue	No action, continues playing
Paused	Stop	Stops
	Pause	None
	Play	Plays from beginning
	Continue	Plays from current frame

If an animation has been built with a limited duration, and if the *Animation Loop Mode* parameter is set to One-Shot, the animation will stop automatically upon its completion. The IG will indicate this condition by sending an **Animation Stop Notification** packet (Section 4.2.15) to the Host. If the *Animation Loop Mode* parameter is set to Loop, the animation will immediately restart from the beginning and no **Animation Stop Notification** packet will be sent.

Once an **Entity Control** packet describing an entity is sent to the IG, the state of that entity will not change until another **Entity Control** packet specifying that entity ID is received. For example, packets describing the Ownship and a wingman 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 Control** packet are as follows:



*1 *Entity State*
*2 *Attach State*
*3 *Collision Detection Enable*
*4 *Inherit Alpha*
*5 *Ground/Ocean Clamp*
*6 Reserved
*7 *Animation Direction*
*8 *Animation Loop Mode*
*9 *Animation State*
*10 *Linear Extrapolation/Interpolation Enable*

Figure 42 – Entity Control 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 – Entity Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 2	This parameter identifies this data packet as the **Entity Control** packet. The value of this parameter must be 2.
Packet Size Type: unsigned int8 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 48.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which this packet is applied. A value of zero (0) corresponds to the Ownship.
Entity State Type: unsigned 2-bit field Units: N/A Values: 0 Inactive/Standby 1 Active 2 Destroyed	This parameter specifies whether the entity should be active or destroyed. This parameter may be set to one of the following values: **Inactive/Standby** – The entity is loaded, but its tree is not enabled in the scene graph. The entity is invisible, and no transformations are applied. Additionally, the entity is excluded from line of sight and collision testing. **Active** – The entity's tree is enabled in the scene graph. Transformations are applied to the entity and it is included in line of sight and collision testing. **Destroyed** – The entity's tree is removed from the scene graph. Any children are also destroyed. All other parameters in this packet are ignored.
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the entity should be attached as a child to a parent. If this parameter is set to Detach (0), the entity becomes or remains a top-level (non-child) entity. The *Parent ID* parameter is ignored. The *Yaw*, *Pitch*, *Roll*, *Latitude*, *Longitude*, and *Altitude* parameters all specify the entity's position relative to the geodetic coordinate system (see Section 3.4.1). If this parameter is set to Attach (1), the entity becomes or remains attached to the entity specified by the *Parent ID* parameter. The parent must already exist, having been created in a prior frame or earlier in the current frame. The *Yaw*, *Pitch*, *Roll*, *X Offset*, *Y Offset*, and *Z Offset* parameters all specify the entity's position relative to the parent's coordinate system (see Section 3.4.2). This parameter may be changed for a given entity at any time. The attachment or detachment takes place immediately and remains in effect until changed with another **Entity Control** packet.

Parameter	Description
Collision Detection Enable Type: 1-bit field Units: N/A Values: 0 Disabled 1 Enabled	This parameter determines whether any collision detection segments and volumes associated with this entity are used as the source in collision testing. If this parameter is set to Enabled (1), every frame each collision detection segment is tested for intersections with polygons not associated with this entity and each collision detection volume is tested pair-wise with every other volume that is not associated with the entity. If this parameter is set to Disabled (0), any collision detection segments defined for the entity are ignored and any collision detection volumes are only tested (as the destination) against volumes defined for entities whose collision detection is enabled. See Sections 4.1.22 and 4.1.23 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 is combined with the apparent alpha of its parent. The following formula will be used to combine the alpha values: $$\alpha = \frac{\alpha_0 \cdot \alpha_p}{255}$$ 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. Note that a change in an entity's alpha affects the entities below it in the hierarchy if those entities inherit their parents' alphas. If *Attach State* is set to Detach (0), this parameter is ignored.

Parameter	Description
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 should be clamped to the ground or water surface. If *Attach State* is set to Attach (1), this parameter is ignored. **No Clamp** – The entity is not clamped. The *Altitude* parameter specifies the entity's height above Mean Sea Level. The *Pitch* and *Roll* parameters specify the entity's pitch and roll relative to the geodetic reference plane (Figure 18, page 25). **Non-Conformal** – The entity is clamped. The *Altitude* parameter specifies an offset above the terrain or sea level. The *Pitch* and *Roll* parameters specify the entity's pitch and roll relative to the geodetic reference plane (Figure 18, page 25). **Conformal** – The entity is clamped and its attitude conforms to the terrain. The *Altitude* parameter specifies an offset above the terrain or sea level. The *Pitch* and *Roll* parameters specify the entity's pitch and roll relative to the slope of the terrain or water.
Animation Direction Type: 1-bit field Units: N/A Values: 0 Forward 1 Backward	This parameter specifies the direction in which an animation plays. This parameter applies only when the value of the *Entity Type* parameter corresponds to an animation.
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.

Parameter	Description
Animation State Type: unsigned 2-bit field Units: N/A Values: 0 Stop 1 Pause 2 Play 3 Continue	This parameter specifies the state of an animation. This parameter applies only when the value of the *Entity Type* parameter corresponds to an animation. **Stop** – Stops the animation sequence. If the animation has a termination sequence or decay behavior, the animation will switch to that behavior. Has no effect if the animation is currently stopped. **Pause** – Pauses playback of an animation. The entity's geometry will remain visible, provided *Entity State* is set to Active. **Play** – Begins or restarts playback from the first animation frame. **Continue** – Continues a playing animation from the current frame of the animation sequence. If the animation is paused, playback is resumed from the current frame. If the animation is stopped, playback is restarted from the first frame of the sequence.
Linear Extrapolation/Interpolation Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the entity's motion may be smoothed by extrapolation or interpolation algorithms on the IG. 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 *Extrapolation/Interpolation Enable* flag of the **IG Control** packet, then smoothing will not be applied to 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 Type Type: unsigned int16 Units: N/A	This parameter specifies the type for the entity. A value of zero (0) indicates a "null" type with no associated geometry. Such an entity might be used to represent the Ownship or a floating camera. When changing entity types, the Host should first delete the entity by setting the *Entity State* parameter to Deactivate (2) and then recreate the entity and any children during a subsequent frame. If the specified type is undefined, the data packet will be disregarded.

Parameter	Description
Parent ID Type: unsigned int16 Units: N/A	This parameter specifies the parent for the entity. If the *Attach State* parameter is set to Detach (0), this parameter is ignored. The value of this parameter may be changed without first detaching the entity from its existing parent. If the specified parent entity is invalid, no change in the attachment will 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 3.4.1.2) If *Attach State* = 0 and *Ground/Ocean Clamp* = 2: Terrain surface polygon orientation If *Attach State* = 1: Entity reference coordinate system (see Section 3.4.2.2)	This parameter specifies the roll angle of the entity. For top-level entities for which *Ground/Ocean Clamp* is set to No Clamp (0) or Non-Conformal (1), this angle is measured from the reference plane described in Section 3.4.1.2. For top-level entities for which *Ground/Ocean Clamp* is enabled, this angle specifies an angular offset from the terrain surface polygon's orientation. For child entities, roll is measured from the entity's reference plane after yaw and pitch rotations have been applied as described in Section 3.4.2.2.
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 3.4.1.2) If *Attach State* = 0 and *Ground/Ocean Clamp* = 2: Terrain surface polygon orientation If *Attach State* = 1: Entity reference plane (see Section 3.4.2.2)	This parameter specifies the pitch angle of the entity. For top-level entities for which *Ground/Ocean Clamp* is set to No Clamp (0) or Non-Conformal (1), this angle is measured from the reference plane described in Section 3.4.1.2. For top-level entities for which *Ground/Ocean Clamp* is enabled, this angle specifies an angular offset from the terrain surface polygon's orientation. For child entities, pitch is measured with respect to the entity's reference plane after the yaw rotation has been applied as described in Section 3.4.2.2.

Parameter	Description
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 is measured from a line parallel to the Prime Meridian as described in Section 3.4.1.2. For child entities, this parameter specifies a rotation about the child entity's **Z** axis when the child's **X** axis is parallel to the parent's **X** axis as described in Section 3.4.2.2.
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.
X Offset (Child Entities) Type: double float Units: meters Datum: Parent's reference point	For child entities, this parameter specifies the distance from the parent's reference point along its parent's **X** axis.
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.
Y Offset (Child Entities) Type: double float Units: meters Datum: Parent's reference point	For child entities, this parameter specifies the distance from the parent's reference point along its parent's **Y** axis.

Parameter	Description
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 value is the height above Mean Sea Level. If the *Ground/Ocean Clamp* parameter is set to Non-Conformal (1) or Conformal (2), this value specifies an offset above the terrain height or local sea level (see Section 4.1.13) 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 distance from the parent's reference point along its parent's **Z** axis.

4.1.3 Conformal Clamped Entity Control

The **Conformal Clamped Entity Control** parameter is used to set spatial data for conformal, ground- or ocean-clamped entities. This packet is offered as a lightweight alternative to the **Entity Control** packet (Section 4.1.2).

Because the entity type and other necessary attributes are not specified in this packet, it may not be used to instantiate (create) an entity. Before using this packet to manipulate an entity, the Host must first instantiate that entity by sending an **Entity Control** packet and should set the *Ground/Ocean Clamp* parameter to Conformal (2). If a non-existent entity is referenced by a **Conformal Clamped Entity Control** packet, the packet will be ignored.

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

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

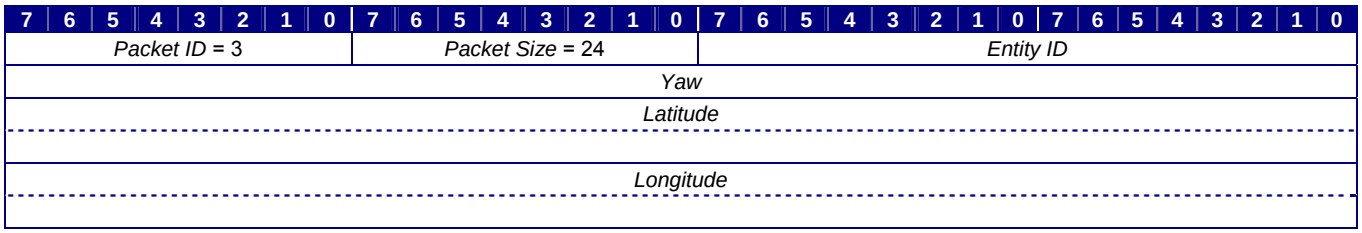


Figure 43 – Conformal Clamped Entity Control Packet Structure

Table 8 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 8 – Conformal Clamped Entity Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 3	This parameter identifies this data packet as the **Conformal Clamped Entity Control** packet. The value of this parameter must be 3.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which this packet is applied. A value of zero (0) corresponds to the Ownship.

Parameter	Description
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 is measured from True North as described in Section 3.4.1.2.
Latitude Type: double float Units: degrees Values: -90 – 90 Datum: Equator	This parameter specifies the entity's geodetic latitude.
Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	This parameter specifies the entity's geodetic longitude.

4.1.4 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 can 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 9 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 9 – 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

Component Class	*Instance ID* Corresponds to	Example *Component ID* Assignments	Example States and Values
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
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 has zero or more discrete states, and/or up to six user-defined values. The user-defined values may either be integers or floating-point real numbers. 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 six user-defined data fields are formatted, they will 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 44 illustrates an incorrect and a correct way of formatting the 32-bit user-defined data fields to store these values:

Incorrect:

Packet ID	*Packet Size*	*Component ID*	
Instance ID		*Component Class*	*Component State*
i		***j***	
k			
m			
0			
0			

Correct:

Packet ID	*Packet Size*	*Component ID*	
Instance ID		*Component Class*	*Component State*
$(i \times 2^{16}) + j$			
k			
***m* (MSW)**			
***m* (LSW)**			
0			
0			

Figure 44 – 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:

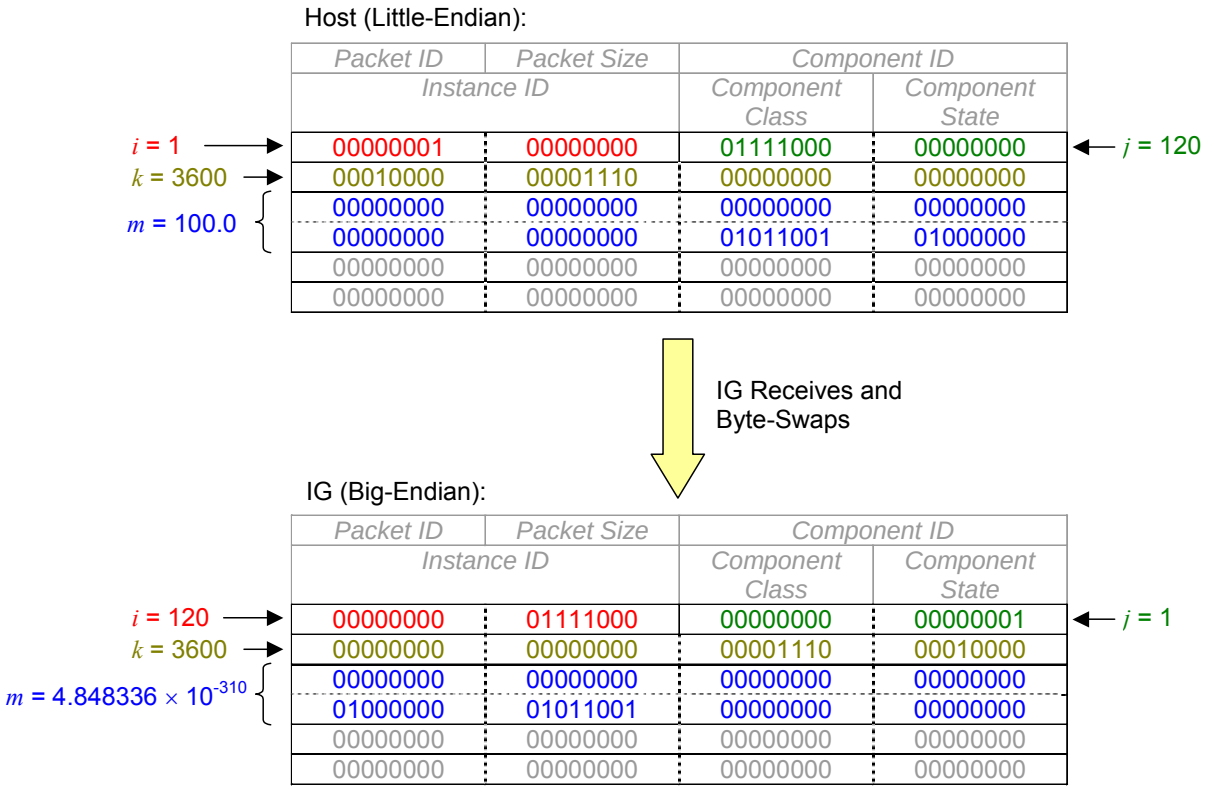


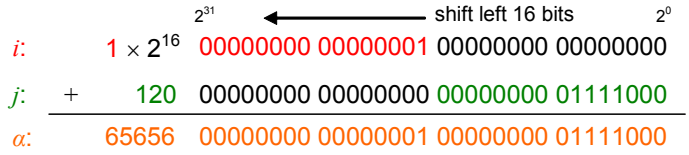
Figure 45 – Example of Incorrect Packaging of Component Data

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.

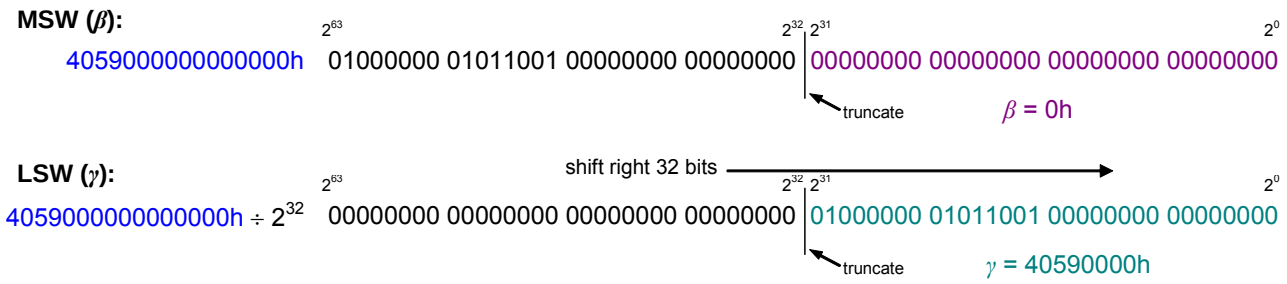
The diagram below illustrates the packing of the data into the correct format shown in Figure 44. Note that to find the least-significant word of m, the variable is typecast as an unsigned 64-bit integer.

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



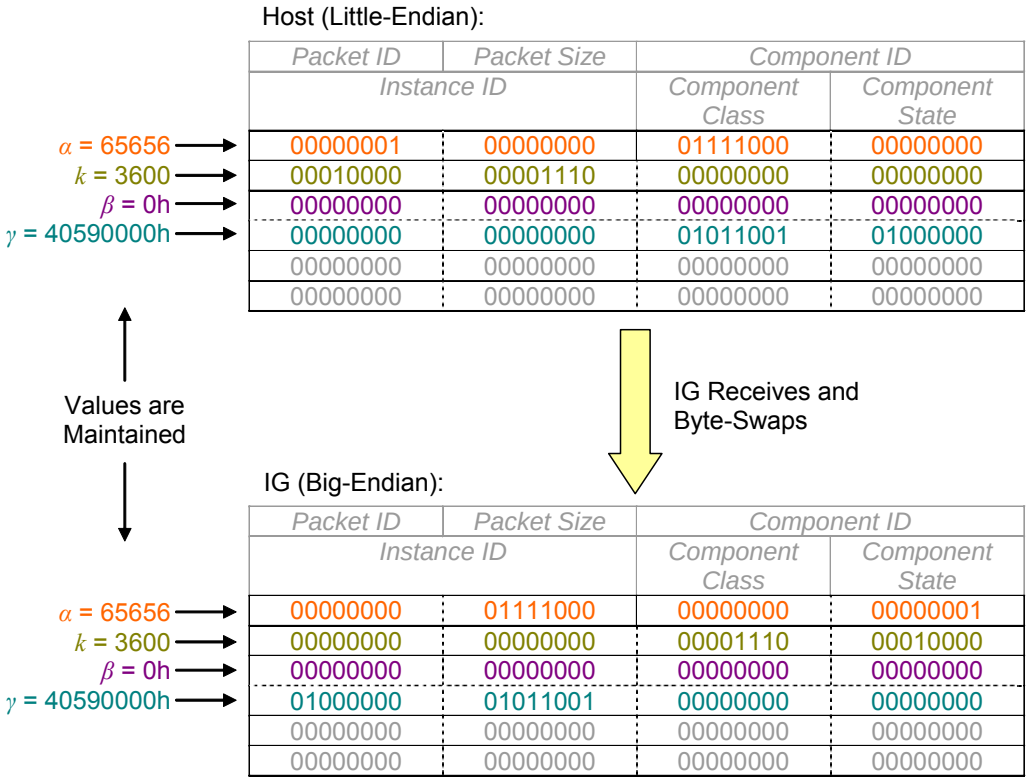


Figure 46 – Example of Correct Packaging of Component Data

k is a 32-bit data type and will be byte-swapped correctly without manipulation. Since α, β, and γ are 32-bit fields, their values will 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 bit fields varies from one compiler to the next, 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 4.1.5). All components that can be controlled with the **Short Component Control** packet can 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 ID = 4								*Packet Size* = 32								*Component ID*															
Instance ID																*1				*Component Class*				*Component State*							
Component Data 1																															
Component Data 2																															
Component Data 3																															
Component Data 4																															
Component Data 5																															
Component Data 6																															

*1 Reserved

Figure 47 – 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 – Component Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 4	This parameter identifies this data packet as the **Component Control** packet. The value of this parameter must be 4.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.

Parameter	Description
Component ID Type: unsigned int16 Units: N/A	This parameter uniquely identifies the component to which the data in this packet should be applied. If *Component Class* is set to Regional Layered Weather (6), the weather layer ID is specified by the most significant byte of *Component ID*.
Instance ID Type: unsigned int16 Units: N/A	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 9). This parameter will 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 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 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 9).
Component State Type: unsigned int8 Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter specifies a discrete state for the component. If a discrete state is not applicable to the component, this parameter is ignored.

Parameter	Description
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.

Parameter	Description
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. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.

4.1.5 Short Component Control

The **Short Component Control** packet, like the **Component Control** packet (Section 4.1.4), is a generic packet 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 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 should be interchangeable. In other words, all components that can be controlled with the **Short Component Control** packet can also be controlled 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). The two packet types can then be processed by the same packet-handling routine.

The *Component Data 1* and *Component Data 2* fields will be byte-swapped, if necessary, as 32-bit data types. Data should be packed into 32-bit units as described in Section 4.1.4.

The contents of the **Short 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 ID = 5								*Packet Size* = 16								*Component ID*															
Instance ID																*1				*Component Class*				*Component State*							
Component Data 1																															
Component Data 2																															

*1 Reserved

Figure 48 – Short Component 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 – Short Component Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 5	This parameter identifies this data packet as the **Short Component Control** packet. The value of this parameter must be 5.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
Component ID Type: unsigned int16 Units: N/A	This parameter uniquely identifies the component to which the data in this packet should be applied. If *Component Class* is set to Regional Layered Weather (6), the weather layer ID is specified by the most significant byte of *Component ID*.
Instance ID Type: unsigned int16 Units: N/A	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 9). This parameter will 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 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 9).
Component State Type: unsigned int8 Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter specifies a discrete state for the component. If a discrete state is not applicable to the component, this parameter is ignored.

Parameter	Description
Component Data 1 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of two 32-bit words used for user-defined component data. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.
Component Data 2 Type: word Units: Component-specific Values: Component-specific Default: IG-configurable	This parameter is one of two 32-bit words used for user-defined component data. If this parameter is not needed by the component, this value is ignored. Note: This parameter will 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 will be performed correctly.

4.1.6 Articulated Part Control

Articulated parts are entity features that can 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 3.4.3. Examples include wing flaps, landing gear, and tank turrets.

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

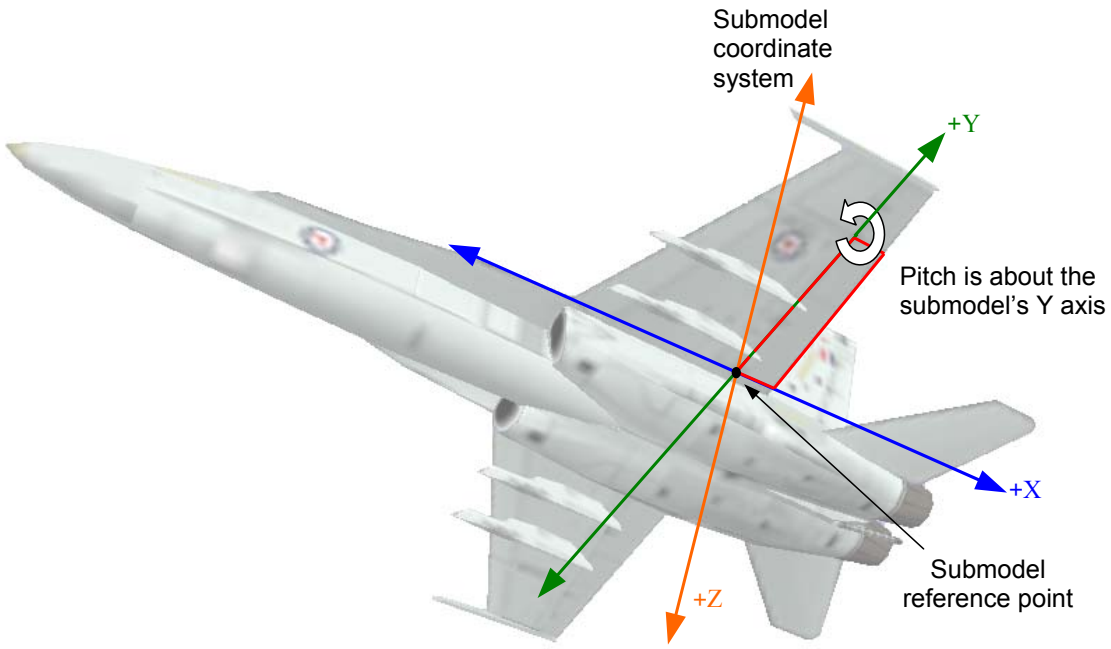


Figure 49 – Manipulation of Articulated Part Submodel

Positional and rotational values are not 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	7	6	5	4	3	2	1	0
Packet ID = 6								*Packet Size* = 32								*Entity ID*															
Articulated Part ID								*8	*7	*6	*5	*4	*3	*2	*1	Reserved															
X Offset																															
Y Offset																															
Z Offset																															
Roll																															
Pitch																															
Yaw																															

[*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 50 – 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 – Articulated Part Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 6	This parameter identifies this data packet as the **Articulated Part Control** packet. The value of this parameter must be 6.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the articulated part belongs.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter specifies the articulated part to which the data in this packet should be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the simulation.

Parameter	Description
Articulated Part Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter determines whether the articulated part submodel should be enabled or disabled within the scene graph. If this parameter is set to Disable (0), the part is removed from the scene; if the parameter is set to Enable (1), the part is included in the scene.
X Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *X Offset* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *X Offset* is ignored and the articulated part remains at its current location along the submodel's **X** axis.
Y Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Y Offset* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *Y Offset* is ignored and the articulated part remains at its current location along the submodel's **Y** axis.
Z Offset Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Z Offset* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *Z Offset* is ignored and the articulated part remains at its current location along the submodel's **Z** axis.
Roll Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Roll* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *Roll* is ignored and the articulated part retains its current roll angle.
Pitch Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Pitch* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *Pitch* is ignored and the articulated part retains its current pitch angle.

Parameter	Description
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Yaw* parameter of the current packet should be applied to the articulated part. If this parameter is set to Disable (0), *Yaw* is ignored and the articulated part retains its current yaw angle.
X Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the distance of the articulated part along its **X** axis.
Y Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the distance of the articulated part along its **Y** axis.
Z Offset Type: single float Units: meters Default: Model-dependent Datum: Submodel reference point	This parameter specifies the distance of the articulated part along its **Z** axis.
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 after yaw and pitch have been applied.

Parameter	Description
Pitch Type: single float Units: degrees Values: -90.0 – 90.0 Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies the angle of rotation of the articulated part submodel about its **Y** axis after yaw has been applied.
Yaw Type: single float Units: degrees Values: 0.0 – 360.0 Default: Model-dependent Datum: Submodel reference coordinate system	This parameter specifies the angle of rotation of the articulated part about its **Z** axis.

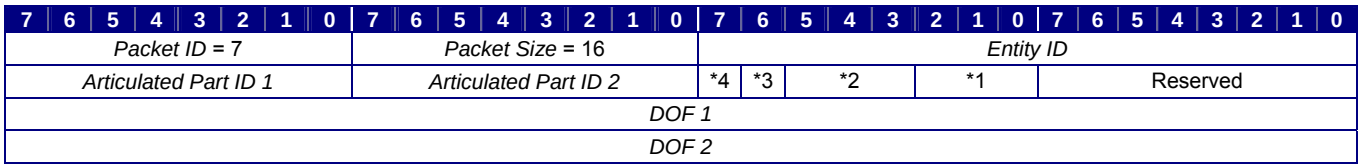
4.1.7 Short Articulated Part Control

The **Short Articulated Part Control** packet is provided as a lower-bandwidth alternative to the **Articulated Part Control** packet (Section 4.1.6). It can 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 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) takes priority over *DOF 1*.

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



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

Figure 51 – Short Articulated Part 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 – Short Articulated Part Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 7	This parameter identifies this data packet as the **Short Articulated Part Control** packet. The value of this parameter must be 7.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the articulated part(s) belongs.
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 should be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the scene graph. 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 should be applied. When used with the *Entity ID* parameter, this parameter uniquely identifies a particular articulated part within the scene graph. 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* is applied. If this parameter is set to Not Used (0), both *DOF 1* and *Articulated Part Enable 1* are 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* is applied. If this parameter is set to Not Used (0), both *DOF 2* and *Articulated Part Enable 2* are ignored.

Parameter	Description
Articulated Part Enable 1 Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter determines whether the articulated part submodel specified by *Articulated Part ID 1* should be enabled or disabled within the scene graph. If this parameter is set to Disable (0), the part is removed from the scene; if the parameter is set to Enable (1), the part is included in the scene.
Articulated Part Enable 2 Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter determines whether the articulated part submodel specified by *Articulated Part ID 2* should be enabled or disabled within the scene graph. If this parameter is set to Disable (0), the part is removed from the scene; if the parameter is set to Enable (1), the part is 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 application 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* specifies an offset in meters. If *DOF Select 1* is set to Yaw (4), Pitch (5), or Roll (6), then *DOF 1* specifies an angular position in 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 application 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* specifies an offset in meters. If *DOF Select 2* is set to Yaw (4), Pitch (5), or Roll (6), then *DOF 2* specifies an angular position in degrees.

4.1.8 Rate Control

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

The **Rate 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.

Rates may also be used to enable the IG to compensate for transport delays or jitter produced by asynchronous operation. A **Rate Control** packet may be sent each frame in conjunction with an **Entity Control** packet. This provides the IG with enough information to extrapolate the entity's probable position during the next frame if necessary.

When a rate is specified for an entity or articulated part, the IG maintains that rate until a new rate is specified by the Host. If the Host changes the position and/or orientation of an entity or articulated part, the IG will perform the transformation and extrapolation will continue from that state beginning with the next frame. If the Host sets all rate components to zero, the entity or articulated part will become stationary.

If the entity to which a rate is applied is destroyed, any rates specified for that entity are annulled.

The contents of the **Rate 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 ID = 8								*Packet Size* = 32								*Entity ID*															
Articulated Part ID								Reserved						*2	*1	Reserved															
X Linear Rate																															
Y Linear Rate																															
Z Linear Rate																															
Roll Angular Rate																															
Pitch Angular Rate																															
Yaw Angular Rate																															

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

Figure 52 – Rate 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 – Rate Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 8	This parameter identifies this data packet as the **Rate Control** packet. The value of this parameter must be 8.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the rate should be applied. If the *Apply to Articulated Part* flag is set to True (1), the rate is applied to an articulated part belonging to this entity. If the flag is set to False (0), the rate is applied to the whole entity.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter specifies the articulated part to which the rate should be applied. If the *Apply to Articulated Part* flag is set to True (1), this parameter refers to an articulated part belonging to the entity specified by *Entity ID*. If the flag is set to False (0), this parameter is ignored.
Apply to Articulated Part Type: 1-bit field Units: N/A Values: 0 False 1 True	This parameter determines whether the rate is applied to the articulated part specified by the *Articulated Part ID* parameter. If this flag is set to False (0), the rate is applied to the entity.
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 rates are applied. When this parameter is set to World/Parent (0) and the entity is a top-level (non-child) entity, the rates are defined relative to the database. Linear rates describe a path along and above the surface of the geoid. Angular rates describe a rotation relative to a reference plane as described in Section 3.3.1.2. When this parameter is set to World/Parent (0) and the entity is a child entity, the rates are defined relative to the parent's local coordinate system as described in Section 3.3.2.2. When this parameter is set to Local (1), the rates are defined relative to the entity's local coordinate system. Note: This parameter is ignored if Apply to Articulated Part is set to True (1)

Parameter	Description
X Linear Rate 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.
Y Linear Rate 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.
Z Linear Rate 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.
Roll Angular Rate 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.

Parameter	Description
Pitch Angular Rate 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.
Yaw Angular Rate 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.

4.1.9 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* 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 *Ephemeris Model Enable* flag.

Note: If the Host freezes the simulation, it must send a **Celestial Sphere Control** packet with the *Ephemeris Model Enable* parameter set to Disable (0); otherwise, the IG will continue to update the time of day. When the Host resumes the simulation, it must explicitly re-enable the ephemeris model.

The *Date/Time Valid* parameter specifies whether the IG should set the current date and time to the values specified by the *Hour*, *Minute*, and *Date* parameters. This enables the Host to change sky model properties without affecting the ephemeris model.

The contents of the **Celestial Sphere 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 ID = 9								*Packet Size* = 16								*Hour*								*Minute*							
Reserved						*5	*4	*3	*2	*1	Reserved																				
Date																															
Star Field Intensity																															

*1 *Ephemeris Model Enable*
*2 *Sun Enable*
*3 *Moon Enable*
*4 *Star Field Enable*
*5 *Date/Time Valid*

Figure 53 – Celestial Sphere 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 – Celestial Sphere Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 9	This parameter identifies this data packet as the **Celestial Sphere Control** packet. The value of this parameter must be 9.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.
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.
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.
Ephemeris Model Enable Type: 1-bit field Units: N/A Values: 0 Disable (Static time of day) 1 Enable (Continuous time of day) Default: 1	This parameter controls whether the time of day is static or continuous. If this parameter is set to Enabled (1), the image generator will continuously update the time of day.
Sun Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the sun is enabled in the sky model.

Parameter	Description
Moon Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the moon is enabled in the sky model. The moon phase is determined by the current date.
Star Field Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the star field is enabled in the sky model. The star positions are determined by the current date and time.
Date/Time Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter specifies whether the *Hour*, *Minute*, and *Date* parameters are valid. If *Date/Time Valid* is set to Valid (1), these values will override the IG's current date and time.
Date Type: unsigned int32 Units: N/A Values: See description at right Default: IG-configurable	This parameter specifies the current date within the simulation. The date is represented as a seven- or eight-digit decimal integer formatted as follows: MMDDYYYY = (month × 1000000) + (day × 10000) + year
Star Field Intensity Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	This parameter specifies the intensity of the star field within the sky model. This parameter is ignored if *Star Field Enable* is set to Disable (0).

4.1.10 Atmosphere Control

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

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

CIGI supports the use of FASCODE, MODTRAN, SEDRIS, 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:

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 ID = 10								*Packet Size* = 32								Reserved								*1	*Global Humidity*						
Global Air Temperature																															
Global Visibility Range																															
Global Horizontal Wind Speed																															
Global Vertical Wind Speed																															
Global Wind Direction																															
Global Barometric Pressure																															
Reserved																															

*1 *Atmospheric Model Enable*

Figure 54 – Atmosphere Control Packet Structure

Table 16 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 16 – Atmosphere Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 10	This parameter identifies this data packet as the **Atmosphere Control** packet. The value of this parameter must be 10.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.

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

Parameter	Description
Global Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	This parameter specifies the global wind direction. Note: This is 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	This parameter specifies the global atmospheric pressure. The units are interchangeable.

4.1.11 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 can be specified. The shape of the region is a rounded rectangle, as shown below:

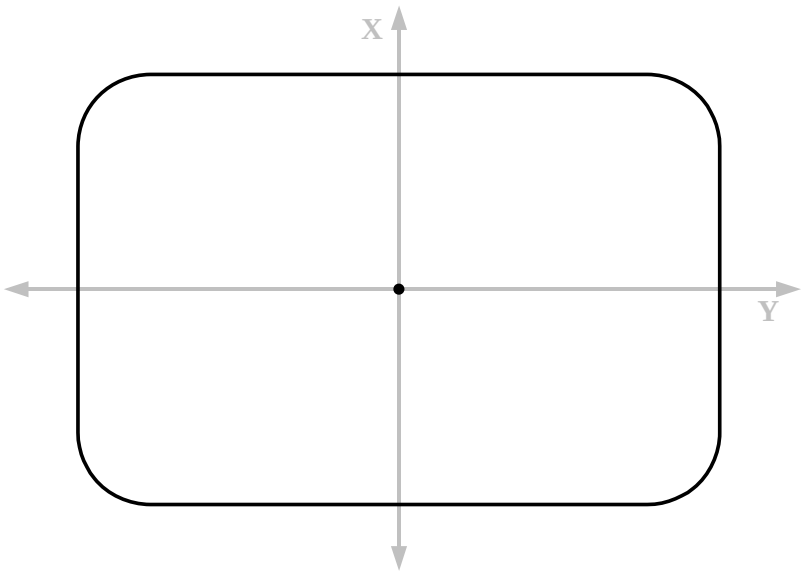


Figure 55 – Example of a Rounded Rectangle on NED Cartesian XY Plane

Up to 256 weather layers may be defined within a region. Weather layers can be created and manipulated with the **Weather Control** packet (Section 4.1.12). Up to 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 does 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 58), respectively.

The "roundness" of the corners is determined by the *Corner Radius* parameter. Setting this radius to zero (0) will create a rectangle. Setting the value equal to one-half that of *Size X* and *Size Y* when both are equal will create a circle. The corner radius must 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 56 shows a rounded rectangle on the NED reference plane rotated by a positive angle ψ.

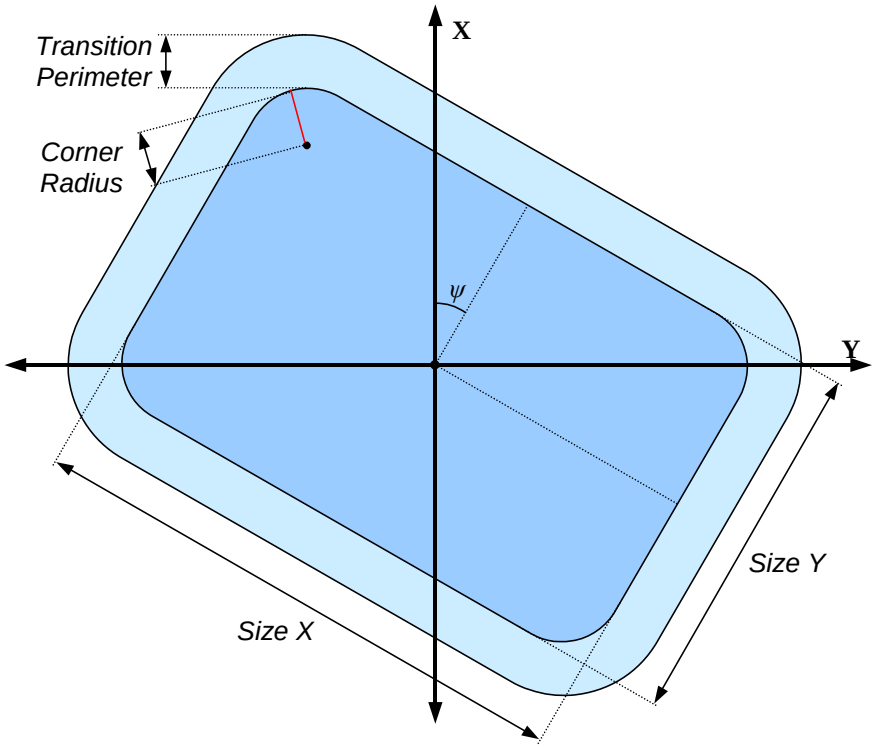


Figure 56 – Rotated Rounded rectangle with Transition Perimeter

The *Transition Perimeter* parameter specifies the width of a corridor around the outside of the rounded rectangle, as illustrated in Figure 56. Within this corridor, the weather conditions will change 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 4.1.12).

To determine the instantaneous value of a weather attribute at a point within the transition perimeter, the IG must 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 could perform linear interpolation to determine the temperature $T_{x,y}$ at some point (x, y) within the transition perimeter.

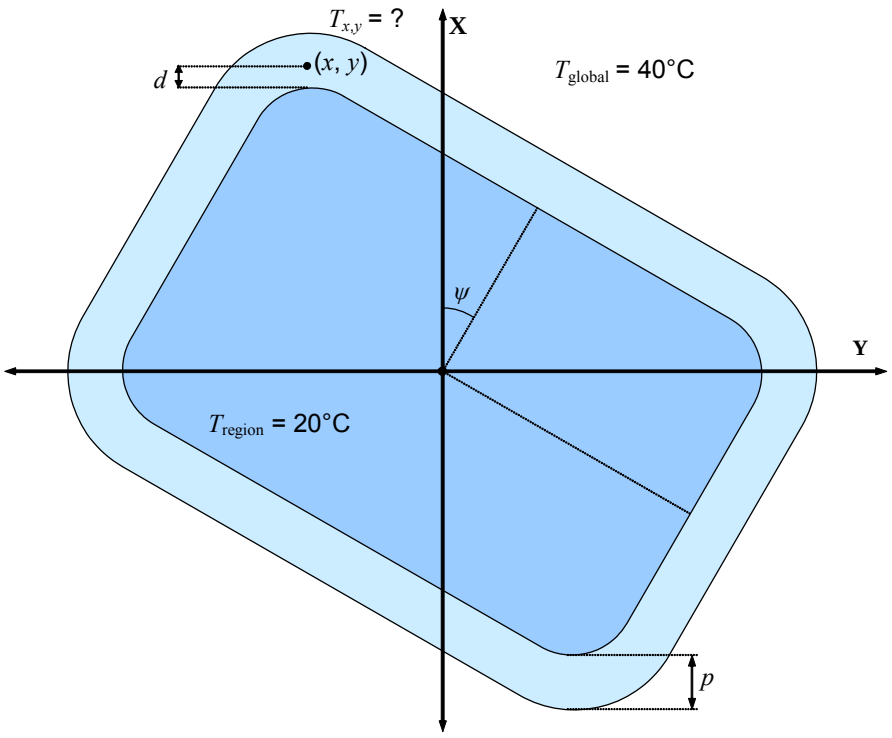


Figure 57 – Interpolation of Temperature within Transition Perimeter

In Figure 57, 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 58). 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 58 shows the rounded rectangle relative to the new coordinate system:

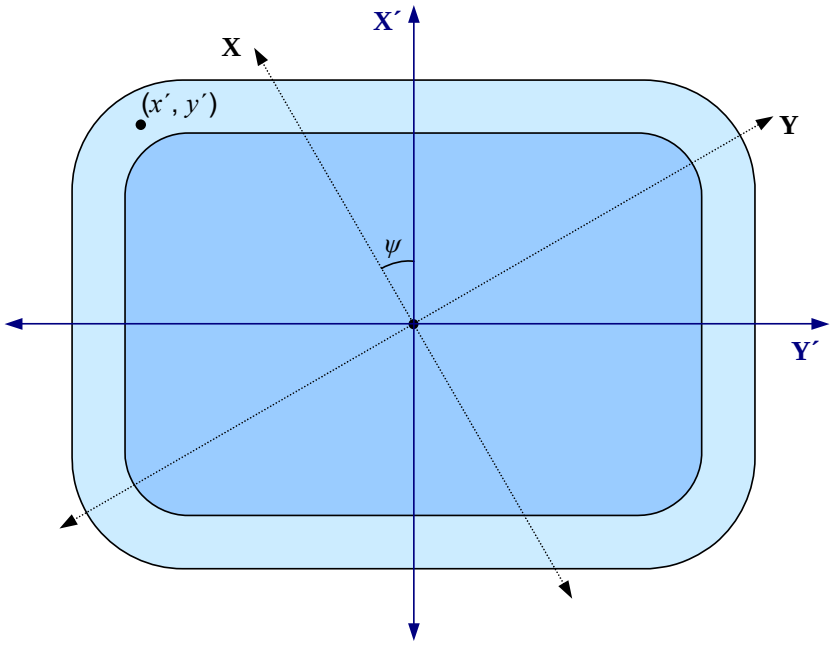


Figure 58 – Rounded Rectangle after Coordinate System Transformation

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 should be 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 59:

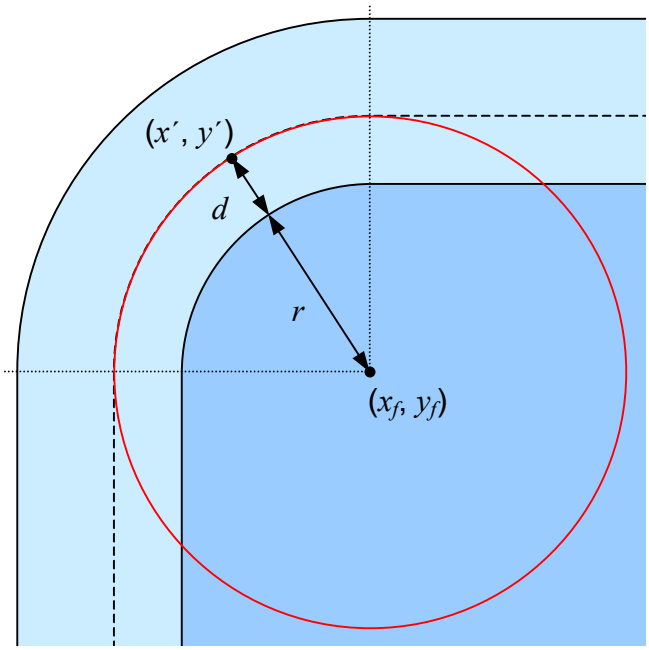


Figure 59 – Circle Drawn though Point (x', y')

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) is given by the following equation:

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

If (x, y) is 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 will overlap layers in other regions. Figure 60 shows two overlapping environmental regions, each containing a cloud layer. The cross-hatched areas indicate the region of overlap.

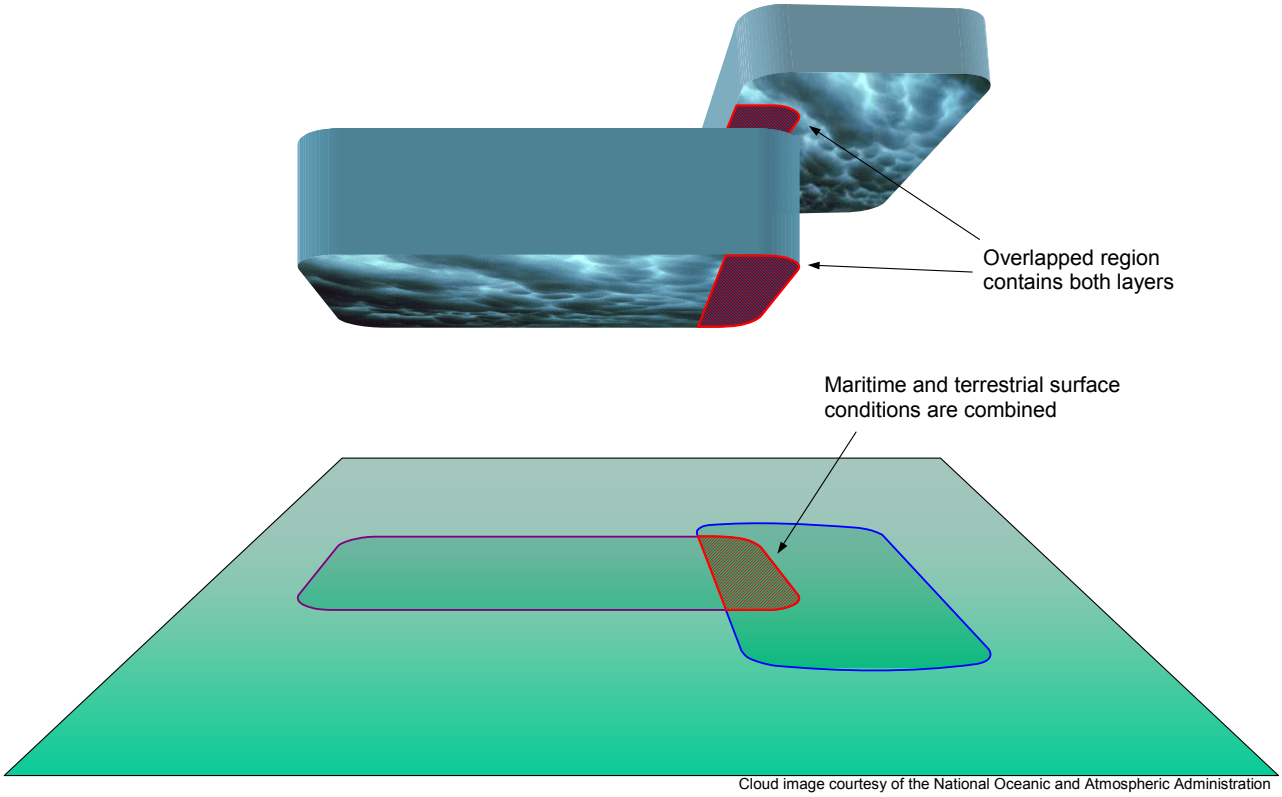


Figure 60 – Example of Overlapping Environmental Regions

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 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 is given to the region whose *Merge Weather Properties* or *Merge Aerosol Concentrations* parameter is set to Merge (1). Table 17 lists the recommended method of combining each property:

Table 17 – 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 17 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 should 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 are each present in their specified concentrations. Any visual or spectral effects caused by the aerosols should be calculated independently for each aerosol.

Maritime and terrestrial surface conditions may also be combined within areas where regions overlap. The recommended method is to average the value of each attribute.

Multiple environmental regions may be arranged to form a weather grid. Figure 61 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.

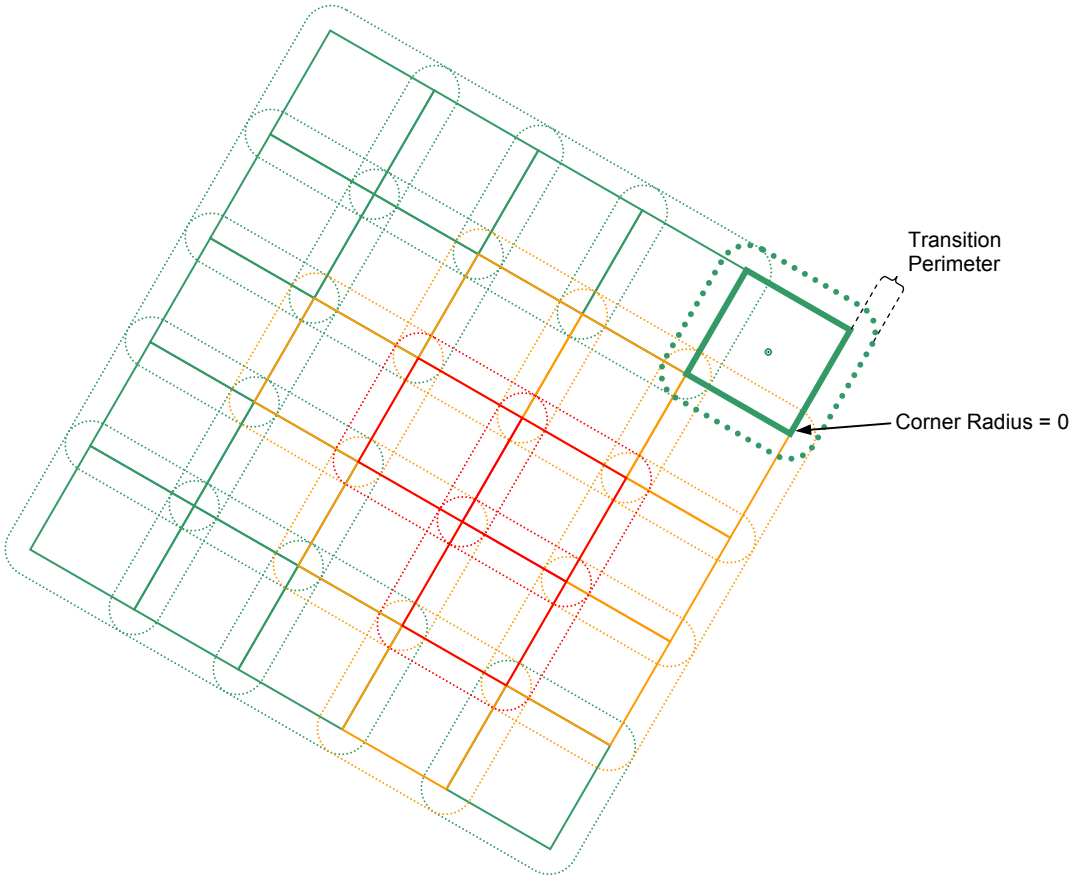


Figure 61 – Example of Gridded Weather System

Circular environmental regions can 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 62 illustrates a group of environmental regions approximating a system of hexagonal weather cells.

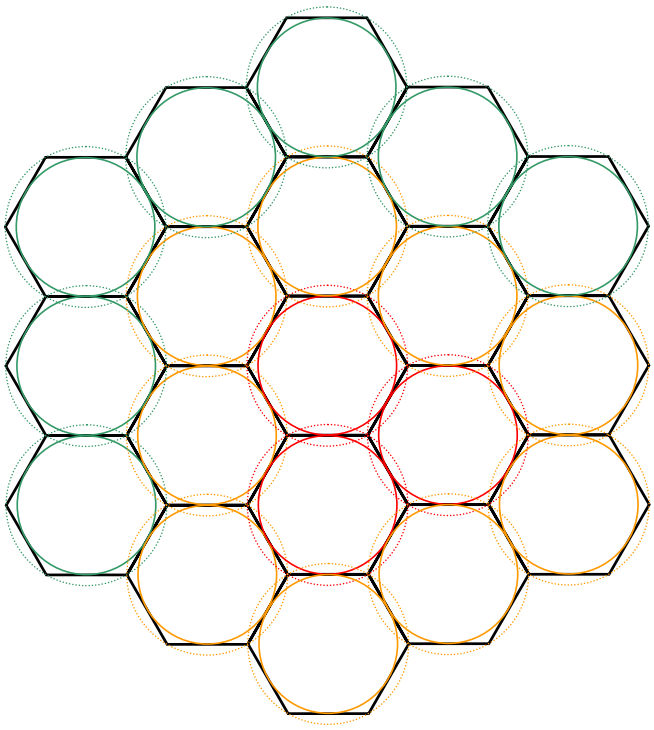


Figure 62 – Example of Approximation of Hexagonal Weather Cells

The *Region State* parameter specifies whether the region should be active, inactive, or destroyed. If the region is inactive, the surface conditions and all weather layers defined within the region are also inactive.

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 ID = 11								*Packet Size* = 48								*Region ID*															
*6		*5		*4		*3		*2		*1		Reserved																			
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 63 – Environmental Region 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 – Environmental Region Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 11	This parameter identifies this data packet as the **Environmental Region Control** packet. The value of this parameter must be 11.
Packet Size Type: unsigned int8 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 48.
Region ID Type: unsigned int16 Units: N/A	This parameter specifies the environmental region to which the data in this packet will be applied.
Region State Type: unsigned 2-bit field Units: N/A Values: 0 Inactive 1 Active 2 Destroyed	This parameter specifies whether the region should be 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 are disabled regardless of their individual enable states. **Active** – Any weather layers and surface conditions defined within the region are enabled according to their individual enable states. **Destroyed** – The environmental region is 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	This parameter specifies whether atmospheric conditions within this region should 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 4.1.12) describing a layer containing a given point will 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 are combined (see Table 17). Note: Weather layers within the same region will 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	This parameter specifies whether the concentrations of aerosols found within this region should 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 4.1.12) describing a layer containing a given point will 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 are combined (see Table 17). Note: Weather layers within the same region will always be combined. Regional weather conditions always take priority over global weather conditions.
Merge Maritime Surface Conditions Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	This parameter specifies whether the maritime surface conditions found within this region should 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 4.1.13) describing a region containing a given point will 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 are averaged with those of any other regions also containing that point. Note: Regional surface conditions always take priority over global surface conditions.

Parameter	Description
Merge Terrestrial Surface Conditions Type: 1-bit field Units: N/A Values: 0 Use Last 1 Merge	This parameter specifies whether the terrestrial surface conditions found within this region should 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 4.1.15) describing a region containing a given point will 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 are averaged with those of any other regions also containing that point. Note: Regional surface conditions always take priority over global surface conditions.
Latitude Type: double float Units: degrees Values: -90.0 – 90.0 Default: IG-configurable Datum: Equator	This parameter specifies the geodetic latitude of the center of the rounded rectangle.
Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Default: IG-configurable Datum: Prime Meridian	This parameter specifies 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	This parameter specifies the length of the environmental region along its X axis at the geoid surface. This length does not include the width of the transition perimeter.

Parameter	Description
Size Y Type: single float Units: meters Values: > 0 Default: IG-configurable Datum: Ellipsoid-tangential reference plane	This parameter specifies the length of the environmental region along its **Y** axis at the geoid surface. This length does 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	This parameter specifies the radius of the corner of the rounded rectangle. The smaller the radius, the "tighter" the corner. A value of 0.0 produces a rectangle.
Rotation Type: single float Units: degrees Values: -180.0 – 180.0 Default: IG-configurable Datum: True North	This parameter specifies the yaw angle of the rounded rectangle.
Transition Perimeter Type: single float Units: meters Values: ≥ 0 Default: IG-configurable	This parameter specifies 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.

4.1.12 Weather Control

The **Weather Control** packet is used to control weather layers and weather entities. Global weather layers have no distinct horizontal boundaries. Atmospheric affects can 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 4.1.11). 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 exists everywhere over the database. If this parameter is set to Region (1), the weather layer is 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 is ignored for weather entities.

The *Cloud Type* parameter specifies the type of cloud found within a cloud layer or entity. Each value may 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 19. The remaining values can 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 64 illustrates the use of these parameters:

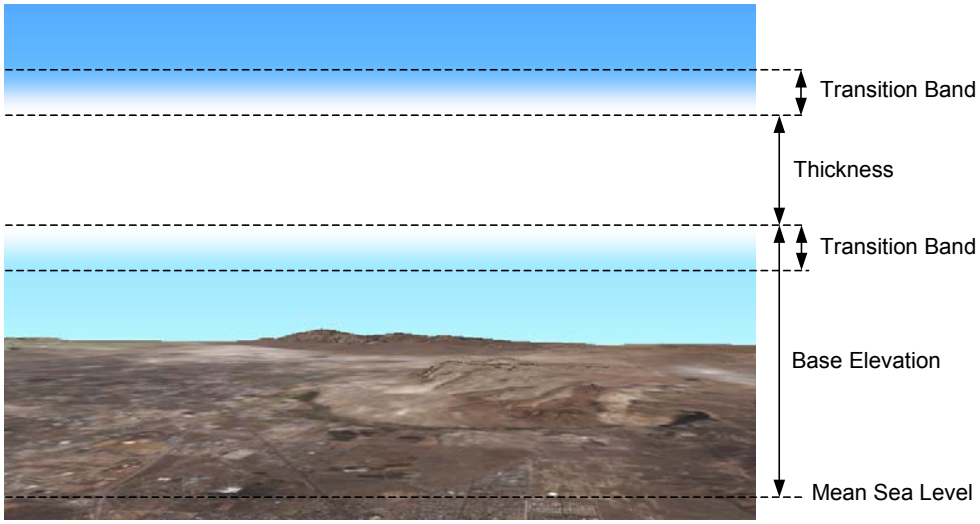


Figure 64 – Weather Layer Base Elevation and Thickness

For weather entities, the *Transition Band* parameter can be used to specify a threshold radius for partial penetration into a cloud model. The *Base Elevation* and *Thickness* parameters are ignored for weather entities.

The *Scud Enable* parameter specifies whether the layer produces scud effects within the transition band. The *Scud Frequency* parameter defines how often scud occurs. The placement of scud (i.e., above versus below the layer) depends upon the IG implementation. Some systems allow this to be controlled via a **Component Control** packet.

The *Horizontal Wind Speed*, *Vertical Wind Speed*, and *Wind Direction* parameters define the wind velocity within the weather layer or entity. These can 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 causes 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 can contain zero or one type of mutable aerosol; multiple aerosols in a given space must be implemented as separate weather layers.

Where weather layers overlap, atmospheric effects should be combined as shown in Table 17 (page 93).

The contents of the **Weather 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 ID = 12								*Packet Size* = 56								*Entity ID/Region ID*															
Layer ID								*Humidity*								*5				*4	*3	*2	*1	Reserved				*7			*6
Air Temperature																															
Visibility Range																															
Scud Frequency																															
Coverage																															
Base Elevation																															
Thickness																															
Transition Band																															
Horizontal Wind Speed																															
Vertical Wind Speed																															
Wind Direction																															
Barometric Pressure																															
Aerosol Concentration																															

*1 *Weather Enable*
*2 *Scud Enable*
*3 *Random Winds Enable*
*4 *Random Lightning Enable*
*5 *Cloud Type*
*6 *Scope*
*7 *Severity*

Figure 65 – Weather 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 – Weather Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 12	This parameter identifies this data packet as the **Weather Control** packet. The value of this parameter must be 12.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 56	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 56.
Entity ID (Weather Entities) Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the weather attributes in this packet are applied.
Region ID (Regional Layers) Type: unsigned int16 Units: N/A	This parameter specifies the region to which the weather layer is confined. Note: *Entity ID/Region ID* is ignored if *Scope* is set to Global (0).
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	This parameter specifies 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 is ignored if *Scope* is set to Entity (2).
Humidity Type: unsigned int8 Units: percent Values: 0 – 100 Default: IG-configurable	This parameter specifies the humidity within the weather layer/entity.

Parameter	Description
Weather Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether a weather layer/entity and its atmospheric effects are enabled.
Scud Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether weather layer produces scud effects within its transition bands.
Random Winds Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether a random frequency and duration should 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	This parameter specifies whether the weather layer or entity exhibits random lightning effects. The frequency and severity of the lightning varies according to the *Severity* parameter.

Parameter	Description
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	This parameter specifies the type of clouds contained within the weather layer. If the value of *Layer ID* does not correspond to a cloud layer, *Cloud Type* should be set to None (0).
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	This parameter specifies whether the weather is global, regional, or assigned to an entity. If this value is set to Regional (1), the layer is confined to the region specified by *Region ID*. If this value is set to Entity (2), the weather attributes are applied to the volume within the moving model specified by *Entity ID*.
Severity Type: unsigned 3-bit field Units: N/A Values: 0 – 5 (Least to most severe) Default: IG-configurable	This parameter specifies the severity of the weather layer/entity.
Air Temperature Type: single float Units: degrees Celsius (°C) Default: IG-configurable	This parameter specifies the temperature within the weather layer/entity.

Parameter	Description
Visibility Range Type: single float Units: meters Values: See description at right Default: IG-configurable	This parameter specifies 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 takes precedence over that specified by the *Global Visibility Range* parameter of the **Atmosphere Control** packet (see Section 4.1.10).
Scud Frequency Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	This parameter specifies the frequency of scud within the transition bands above and/or below a cloud or fog layer. A value of 0% produces no scud effect; 100% produces a solid effect.
Coverage Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	This parameter specifies the amount of area coverage for the weather layer.
Base Elevation Type: single float Units: meters Default: IG-configurable Datum: Mean Sea Level	This parameter specifies the altitude of the base (bottom) of the weather layer. This parameter is ignored if *Scope* is set to Entity (2).
Thickness Type: single float Units: meters Default: IG-configurable Datum: Base (specified by *Base Elevation*)	This parameter specifies 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 is ignored if *Scope* is set to Entity (2).
Transition Band Type: single float Units: meters Default: IG-configurable	This parameter specifies the height of a vertical transition band both above and below the weather layer. This band produces a visibility gradient from the layer's visibility to that immediately outside the transition band. This parameter is ignored if *Scope* is set to Entity (2).

Parameter	Description
Horizontal Wind Speed Type: single float Units: m/s Values: ≥ 0 Default: 0	This parameter specifies the local wind speed parallel to the ellipsoid-tangential reference plane.
Vertical Wind Speed Type: single float Units: m/s Default: 0	This parameter specifies the local vertical wind speed. Note: A positive value produces an updraft, while a negative value produces a downdraft.
Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	This parameter specifies the local wind direction. Note: This is the direction from which the wind is blowing.
Barometric Pressure Type: single float Units: millibars (mb) or hectopascals (hPa) Values: ≥ 0 Default: IG-configurable	This parameter specifies 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	This parameter specifies 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.

4.1.13 Maritime Surface Conditions Control

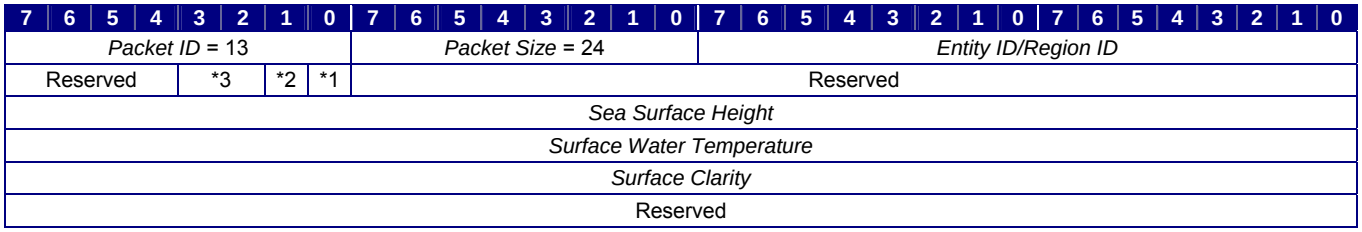
The **Maritime Surface Conditions Control** packet is used to specify the surface behavior for seas and other bodies of water. This packet is used in conjunction with the **Weather Control** and **Wave Control** packets to define sea states.

Regional maritime surface conditions always take precedence over the global surface conditions. Once the surface conditions of a region are set, global changes will not affect the surface conditions within that region unless it is disabled. Global changes will, 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 should 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 can request the conditions at a specific latitude and longitude by issuing an **Environmental Conditions Request** packet (Section 4.1.28). The Host can request the instantaneous height of the water surface at a specific latitude and longitude by sending a **HAT/HOT Request** packet (Section 4.1.24).

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



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

Figure 66 – Maritime Surface Conditions 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 – Maritime Surface Conditions Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 13	This parameter identifies this data packet as the **Maritime Surface Conditions Control** packet. The value of this parameter must be 13.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
Entity ID (Entity-based Surface Conditions) Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the surface attributes in this packet are applied.
Region ID (Regional Surface Conditions) Type: unsigned int16 Units: N/A	This parameter specifies the region to which the surface attributes are confined. Note: *Entity ID/Region ID* is ignored if *Scope* is set to Global (0).
Surface Conditions Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	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 are the same as the global maritime surface conditions. If the parameter is set to Enable (1), the surface conditions are defined by this packet. This parameter is 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	This parameter determines whether whitecaps are enabled.
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	This parameter specifies whether this packet is applied globally, applied to a region, or assigned to an entity. If this value is set to Regional (1), the surface condition properties are applied only within the region specified by *Region ID*. If this value is set to Entity (2), the properties are applied to the area defined by the moving model specified by *Entity ID*.

Parameter	Description
Sea Surface Height Type: single float Units: meters Datum: Mean Sea Level Default: IG-configurable	This parameter specifies the height of the water above MSL at equilibrium. This parameter can 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	This parameter specifies the water temperature at the surface.
Surface Clarity Type: single float Units: percent Values: 0 – 100 Default: IG-configurable	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% indicates pristine water. A value of 0% indicates extremely turbid water.

4.1.14 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:

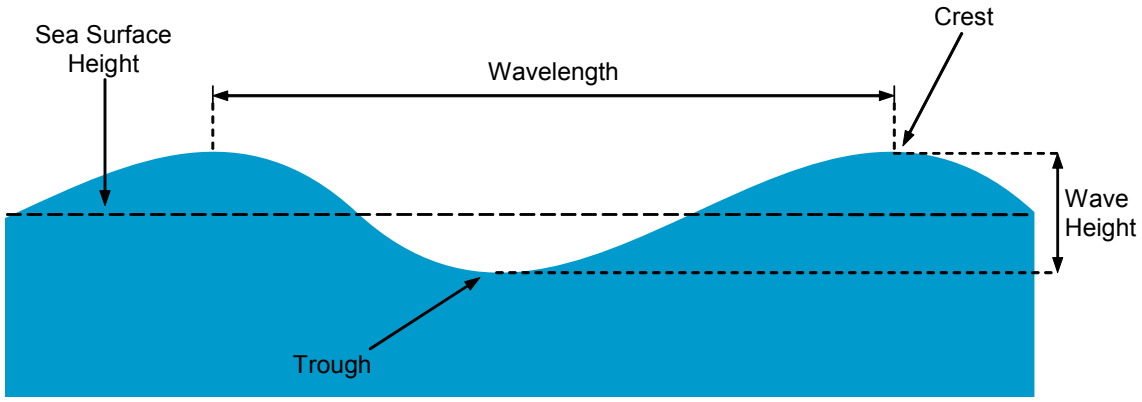


Figure 67 – Basic Wave Properties

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:

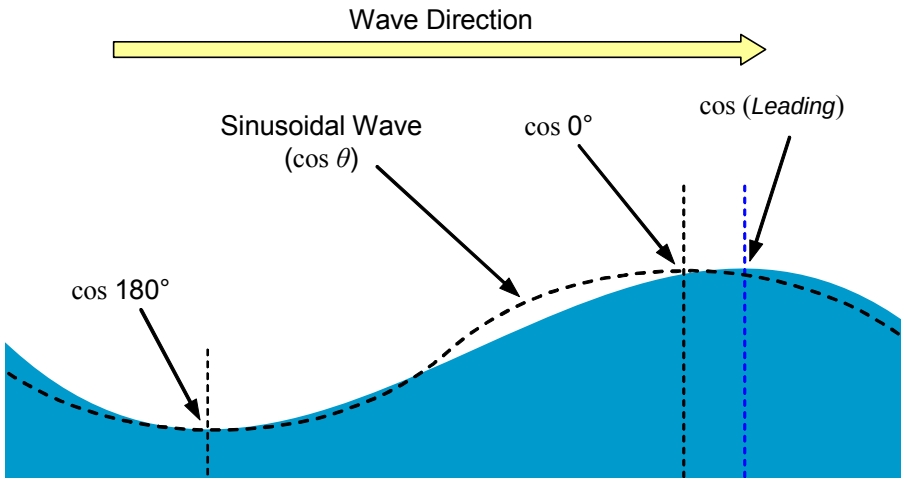


Figure 68 – Example of Wave Leading

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 ID = 14								*Packet Size* = 32								*Entity ID/Region ID*															
Wave ID								Reserved				*3		*2		*1	Reserved														
Wave Height																															
Wavelength																															
Period																															
Direction																															
Phase Offset																															
Leading																															

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

Figure 69 – Wave 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 – Wave Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 14	This parameter identifies this data packet as the **Wave Control** packet. The value of this parameter must be 14.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
Wave Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter determines whether the wave is enabled or disabled. A disabled wave does 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	This parameter specifies whether the wave is defined for global, regional, or entity-controlled maritime surface conditions. If this value is set to Regional (1), the wave properties are applied only within the region specified by *Region ID*. If this value is set to Entity (2), the properties are applied to the area defined by the moving model specified by *Entity ID*.
Breaker Type Type: unsigned 2-bit field Units: N/A Values: 0 Plunging 1 Spilling 2 Surging Default: IG-configurable	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 will then break violently into the wave trough. **Spilling** – Spilling breakers break gradually over a great distance. White water forms 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 does not fall over the front of the wave.
Entity ID (Entity-based Surface Conditions) Type: unsigned int16 Units: N/A	This parameter specifies the surface entity for which the wave is defined.
Region ID (Regional Surface Conditions) Type: unsigned int16 Units: N/A	This parameter specifies the environmental region for which the wave is defined. Note: *Entity ID/Region ID* is ignored if *Scope* is set to Global (0).
Wave ID Type: unsigned int8 Units: N/A	This parameter specifies the wave to which the attributes in this packet are applied.
Wave Height Type: single float Units: meters Values: ≥ 0 Datum: *Sea Surface Height*	This parameter specifies the average vertical distance from trough to crest produced by the wave.

Parameter	Description
Wavelength Type: single float Units: meters Values: > 0	This parameter specifies the distance from a particular phase on a wave to the same phase on an adjacent wave.
Period Type: single float Units: seconds Values: > 0	This parameter specifies the time required for one complete oscillation of the wave.
Direction Type: single float Units: degrees Values: 0 – 360 Datum: True North	This parameter specifies the direction in which the wave propagates.
Phase Offset Type: single float Units: degrees Values: -360.0 – 360.0	This parameter specifies 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 68).

4.1.15 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.

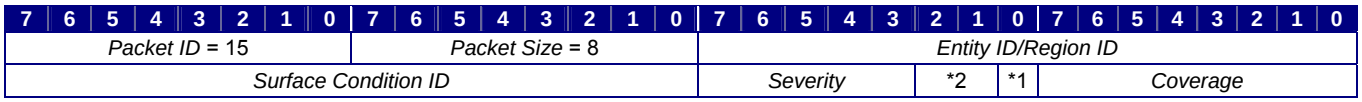
The possible surface conditions are IG-dependent. Examples might range from weather-related conditions such as dry, wet, icy, or slushy, to hazards such as sand, dirt, and gravel.

Regional terrestrial surface conditions always take precedence over the global surface conditions. Once the surface conditions of a region are set, global changes will not affect the surface conditions within that region unless it is disabled. Global changes will, 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 should 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 can request the conditions at a specific latitude and longitude by issuing an **Environmental Conditions Request** packet (Section 4.1.28).

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



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

Figure 70 – Terrestrial Surface Conditions 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 – Terrestrial Surface Conditions Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 15	This parameter identifies this data packet as the **Terrestrial Surface Conditions Control** packet. The value of this parameter must be 15.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.

Parameter	Description
Entity ID (Environmental Entities) Type: unsigned int16 Units: N/A	This parameter specifies the environmental entity to which the surface condition attributes in this packet are applied.
Region ID (Regional Conditions) Type: unsigned int16 Units: N/A	This parameter specifies the region to which the surface conditions are confined. Note: *Entity ID/Region ID* is ignored if *Scope* is set to Global (0).
Surface Condition ID Type: unsigned int16 Units: N/A Values: 0 Dry (reset) > 0 Defined by IG Default: IG-configurable	This parameter identifies a surface condition or contaminant. Multiple conditions can be specified by sending multiple **Terrestrial Surface Conditions Control** packets. When this parameter is set to Dry (0), all existing surface conditions will be removed within the specified scope. All other surface condition codes are IG-dependent.
Surface Condition Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the surface condition attribute identified by the *Surface Condition ID* parameter should be enabled.
Scope Type: unsigned 2-bit field Units: N/A Values: 0 Global 1 Regional 2 Entity Default: IG-configurable	This parameter determines whether the specified surface conditions are applied globally, regionally, or to an environmental entity. If this value is set to Regional (1), the conditions are confined to the region specified by *Region ID*. If this value is set to Entity (2), the conditions are 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	This parameter determines the degree of severity for the specified surface contaminant(s). A value of zero (0) indicates that any effects of the contaminant are negligible. A value of 31 indicates that the surface is not navigable.

Parameter	Description
Coverage Type: unsigned int8 Units: percent Values: 0 – 100 Default: IG-configurable	This parameter determines the degree of coverage of the specified surface contaminant.

4.1.16 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 can 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 is first 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 is first about **Z** axis (yaw), then the **Y** axis (pitch), and finally the **X** axis (roll). Figure 71 illustrates the degrees of freedom for positioning and rotating a view:

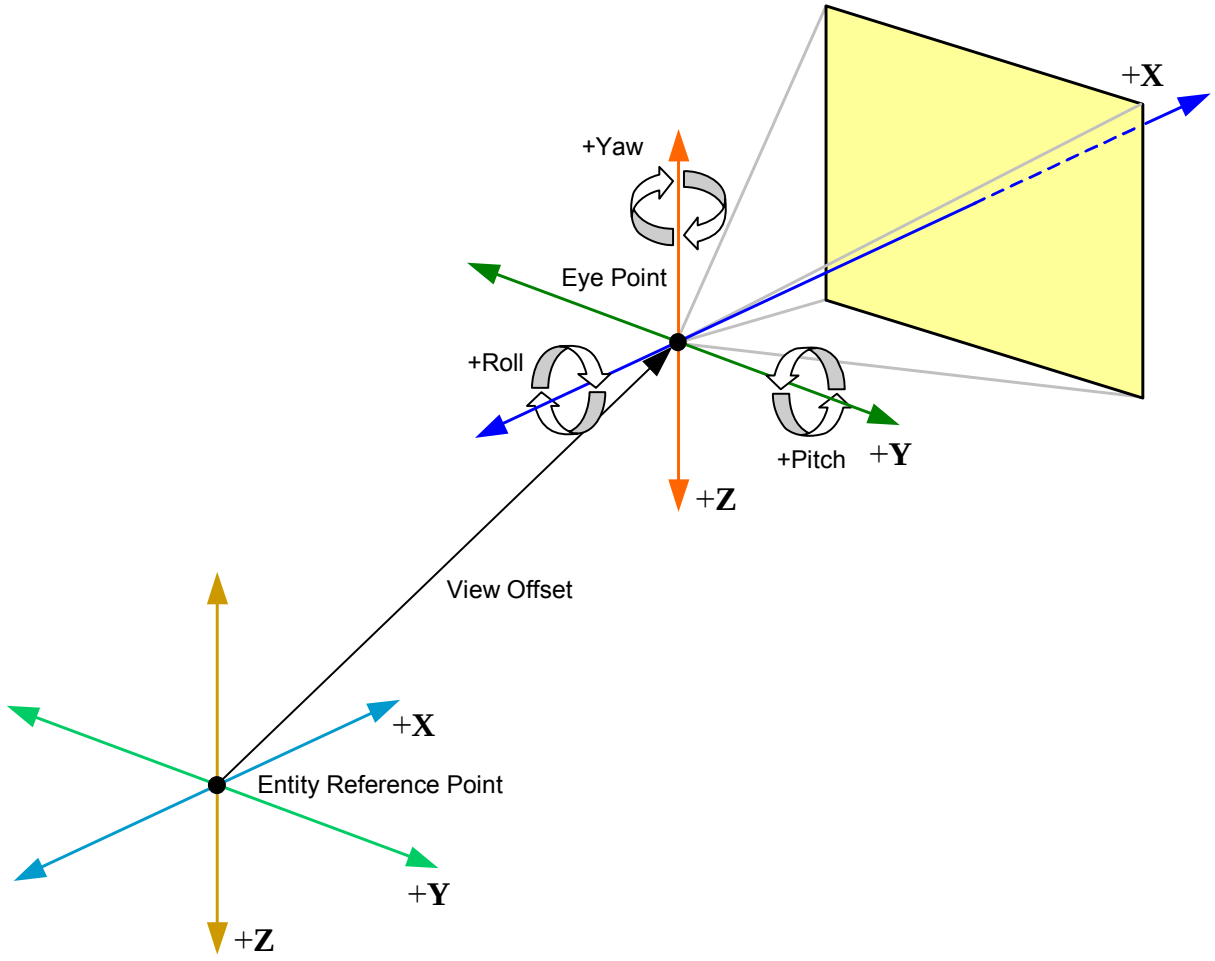


Figure 71 – View Point Position and Rotation

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 ID = 16								*Packet Size* = 32								*View ID*															
Group ID								*7	*6	*5	*4	*3	*2	*1		*Entity ID*															
X Offset																															
Y Offset																															
Z Offset																															
Roll																															
Pitch																															
Yaw																															

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

Figure 72 – View 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 – View Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 16	This parameter identifies this data packet as the **View Control** packet. The value of this parameter must be 16.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
View ID Type: unsigned int16 Units: N/A	This parameter specifies the view to which the contents of this packet should be applied. This value is ignored if the *Group ID* parameter contains a non-zero value.

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

Parameter	Description
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter determines whether the *Yaw* parameter should be applied to the specified view or view group. If this flag is set to Disable (0), the *Yaw* parameter is ignored.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity to which the view or view group should be attached.
X Offset Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies 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	This parameter specifies the position of the view eyepoint along the **Y** axis of the entity specified by the *Entity ID* parameter.
Z Offset Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies 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	This parameter specifies the angle of rotation of the view or view group about its **X** axis after yaw and pitch have been applied.

Parameter	Description
Pitch Type: single float Units: degrees Values: -90 – 90 Default: IG-configurable Datum: View **XY** plane	This parameter specifies 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	This parameter specifies the angle of rotation of the view or view group about its **Z** axis.

4.1.17 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 4.1.16), which moves the sensor camera eyepoint. The **View Definition** and **Component Control** packets (Sections 4.1.21 and 4.1.4, respectively) can 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.

In a typical scenario, the sensor will be inactive until the user 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 will be sent as the user slews the sensor view. This sequence of events is illustrated in Figure 73:

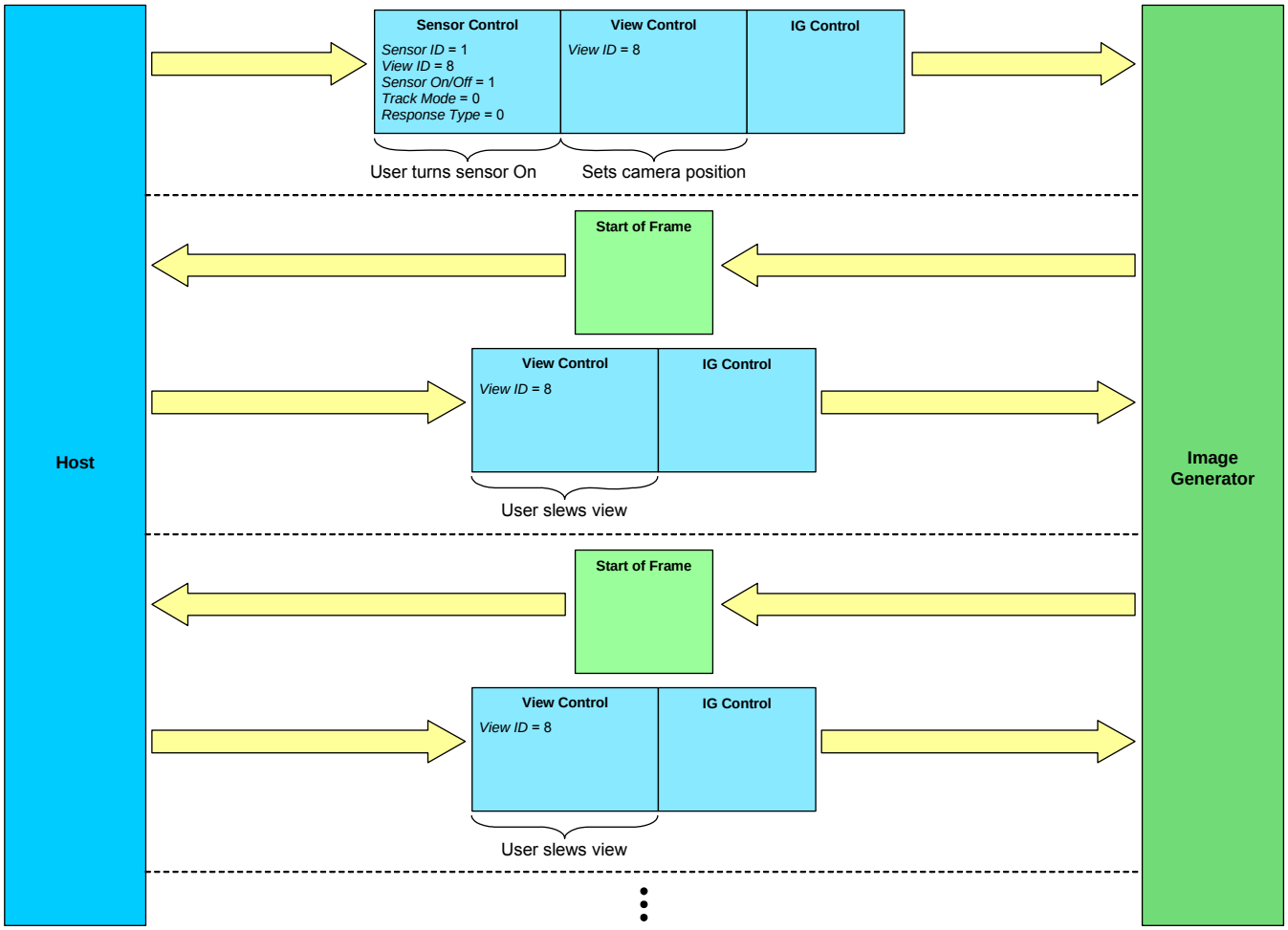


Figure 73 – Data Exchange for Sensor Control (1 of 3)

When the user attempts to lock onto a target, the Host will send a **Sensor Control** packet, setting the *Track Mode* parameter to the appropriate value. Because the Host will need 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 will 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. A response packet will be sent every frame until the IG is directed to do otherwise by the Host.

The *Sensor Status* parameter of the response packets will indicate whether the sensor was able to establish a lock. If the sensor was unable to do so, the *Sensor Status* parameter will 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. If, on the other hand, the lock was successful, then the *Sensor Status* parameter will be set to one (1).

Figure 74 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 that provide gate and target positions.

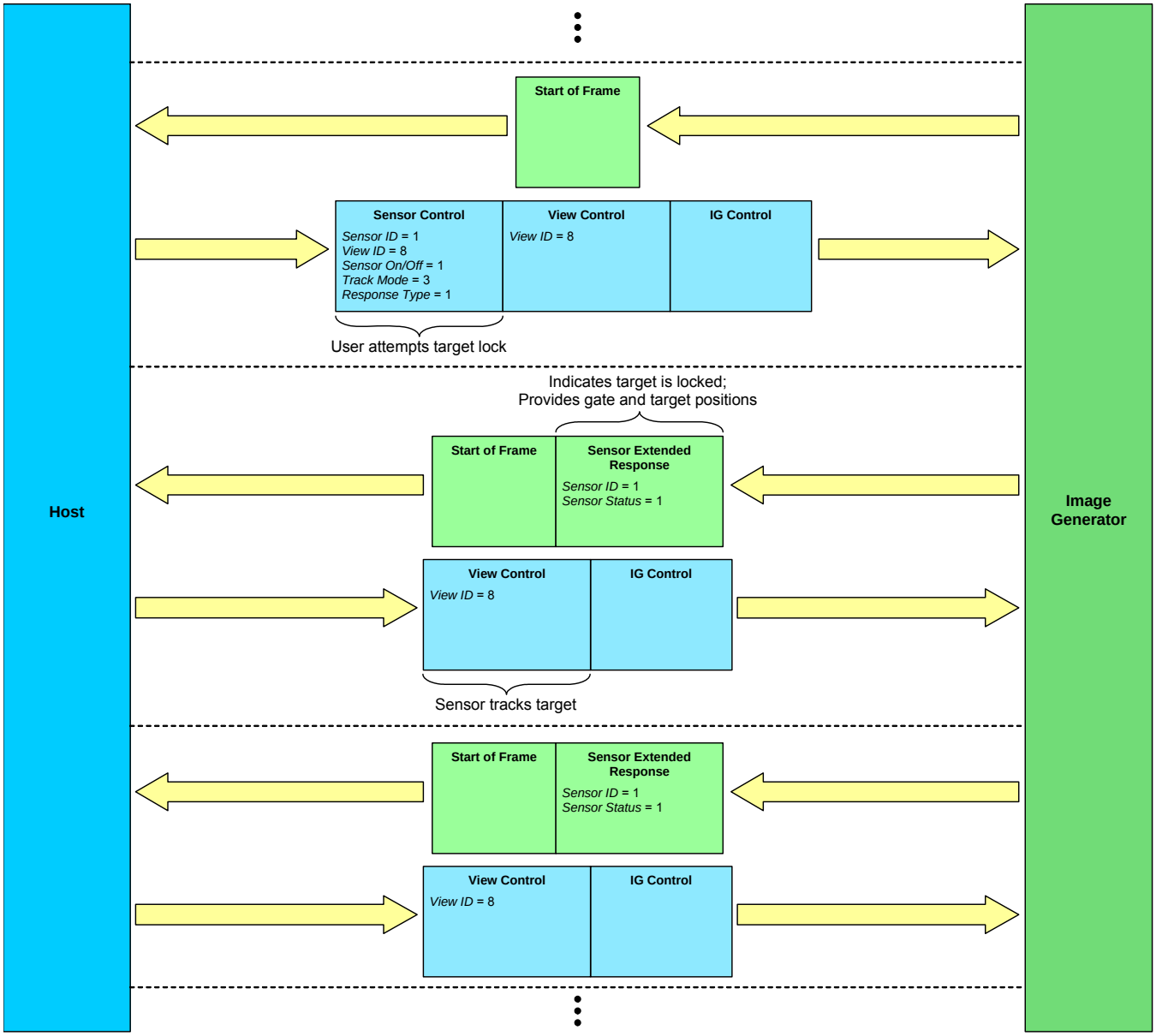


Figure 74 – Data Exchange for Sensor Control (2 of 3)

The *Entity ID* parameter of the **Sensor Extended Response** packet contains the ID of the target entity. If the IG cannot determine the target, or if the sensor is tracking non-entity geometry, then the *Entity ID Valid* parameter of the response packet will be set to Invalid (0). The Host must then use the target position returned by the IG to determine which entity or object is being tracked by the sensor. This may occur immediately or over several frames, depending upon the number and proximity of entities along the sensor viewing vector.

Once the Host has determined the target, it can send a **Sensor Control** packet with its *Response Type* parameter set to Gate Position (1), directing the IG to send **Sensor Response** packets instead of **Sensor Extended Response** packets. This exchange of data is illustrated in Figure 75:

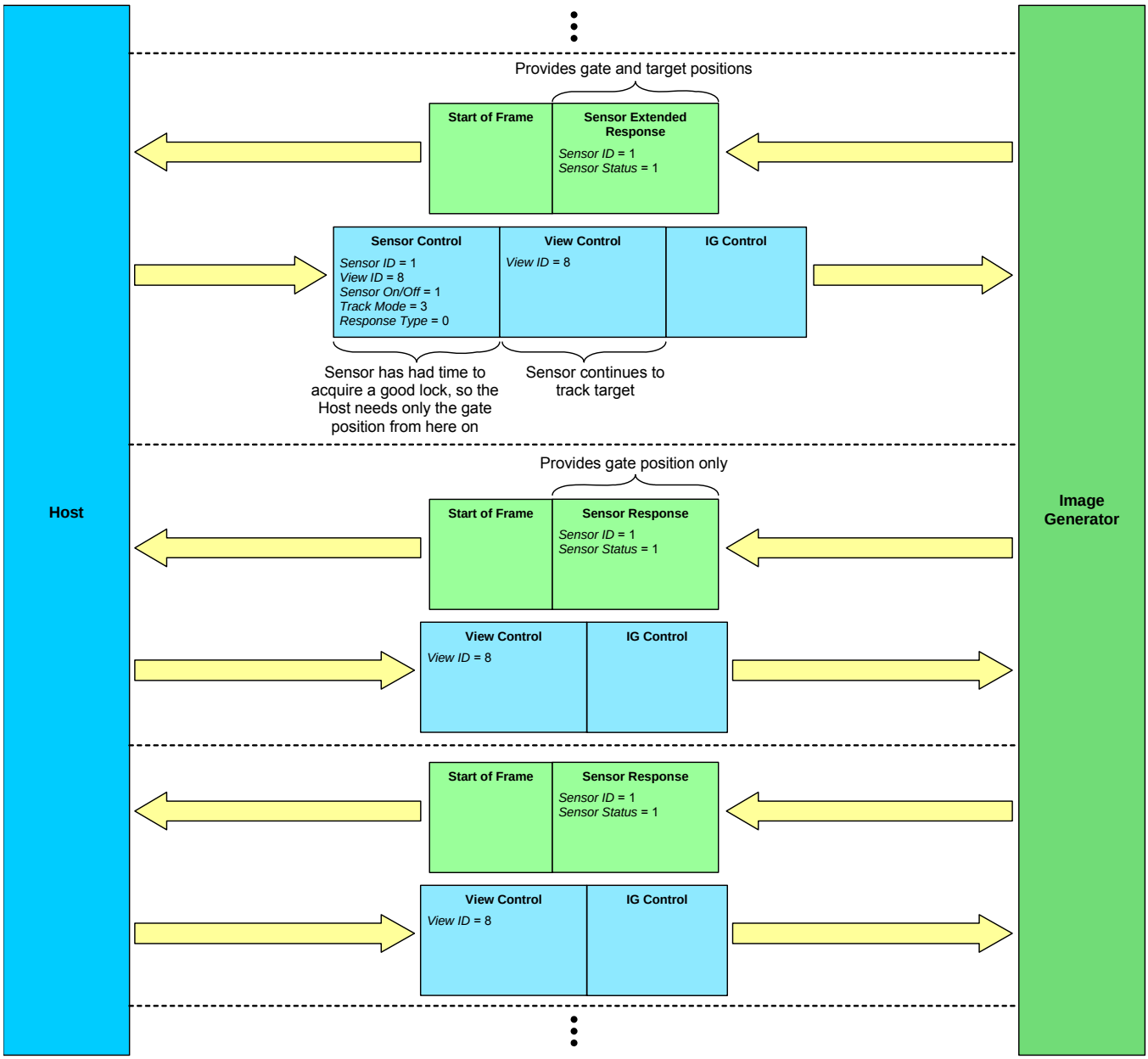


Figure 75 – Data Exchange for Sensor Control (3 of 3)

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 ID = 17								*Packet Size* = 24								*View ID*															
Sensor ID								*Track Mode*		*5		*4		*3		*2		*1	Reserved							*6	Reserved				
Gain																															
Level																															
AC Coupling																															
Noise																															

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

Figure 76 – Sensor 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 – Sensor Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 17	This parameter identifies this data packet as the **Sensor Control** packet. The value of this parameter must be 17.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
View ID Type: unsigned int16 Units: N/A	This parameter identifies the view to which the specified sensor is assigned. Note that a sensor cannot be assigned to a view group.
Sensor ID Type: unsigned int8 Units: N/A	This parameter specifies the sensor to which the data in this packet are 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	This parameter specifies which track mode the sensor should use: **Off** – No tracking will occur. **Force Correlate** – 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. This mode is typically used for Maverick sensors. **Scene** – 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. This mode is typically used for FLIR sensors. **Target** – The sensor uses contrast tracking to lock to a specific target area. **Ship** – The sensor uses contrast tracking and adjusts 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	This parameter specifies whether the sensor is turned on or off.
Polarity Type: 1-bit field Units: N/A Values: 0 White hot 1 Black hot Default: 0	This parameter specifies 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	This parameter specifies whether line-by-line dropout is enabled.

Parameter	Description
Automatic Gain Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	This parameter specifies 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	This parameter specifies whether the sensor tracks white or black. This, along with the *Polarity* parameter, controls whether the sensor tracks hot or cold spots.
Response Type Type: unsigned 1-bit field Units: N/A Values: 0 Normal (gate position) 1 Extended (gate and target position) Default: 0	This parameter specifies whether the IG should return a **Sensor Response** (Section 4.2.6) packet or a **Sensor Extended Response** packet (Section 4.2.7). The IG should 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).
Gain Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	This parameter specifies the contrast for the sensor display.
Level Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	This parameter specifies the brightness for the sensor display.

Parameter	Description
AC Coupling Type: single float Units: µs Values: ≥ 0.0 Default: 0.0	This parameter specifies the AC coupling decay constant for the sensor display.
Noise Type: single float Units: N/A Values: 0.0 – 1.0 Default: 0.0	This parameter specifies the amount of detector noise for the sensor.

4.1.18 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 is determined by the IG.

The Host may request the instantaneous position and orientation of a tracker device by sending a **Position Request** packet (Section 4.1.27) 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 ID = 18								*Packet Size* = 8								*View/View Group ID*															
Tracker ID								*8	*7	*6	*5	*4	*3	*2	*1	Reserved								*9	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 77 – Motion Tracker Control 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 – Motion Tracker Control Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 18	This parameter identifies this data packet as the **Motion Tracker Control** packet. The value of this parameter must be 18.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.

Parameter	Description
View/View Group ID Type: unsigned int16 Units: N/A Default: 0	This parameter specifies the view or view group to which the tracking device is attached.
Tracker ID Type: unsigned int8 Units: N/A	This parameter specifies 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	This parameter specifies whether the tracking device is enabled.
Boresight Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 0	This parameter is used to 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 must send a **Motion Tracker Control** packet with *Boresight Enable* set to Disable (0) to return the tracker to normal operation. The IG will 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	This parameter is used to enable or disable the X-axis position of the motion tracker.

Parameter	Description
Y Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	This parameter is used to enable or disable the **Y**-axis position of the motion tracker.
Z Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	This parameter is used to enable or disable the **Z**-axis position of the motion tracker.
Roll Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	This parameter is used to enable or disable the roll (**X**-axis rotation) of the motion tracker.
Pitch Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	This parameter is used to enable or disable the pitch (**Y**-axis rotation) of the motion tracker.
Yaw Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: 1	This parameter is used to enable or disable the yaw (**Z**-axis rotation) of the motion tracker.

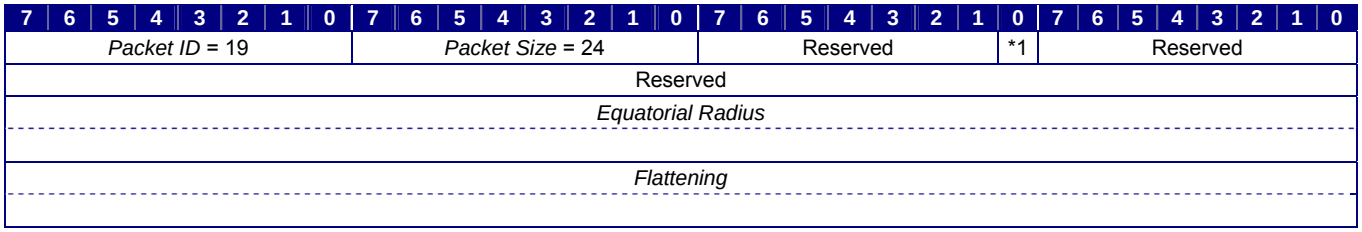
Parameter	Description
View/View Group Select Type: 1-bit field Units: N/A Values: 0 View 1 View Group Default: IG-configurable	This parameter specifies whether the tracking device is attached to a single view or a view group. If set to View (0), the *View/View Group ID* parameter identifies a single view. If set to View Group (1), that parameter identifies a view group.

4.1.19 Earth Reference Model Definition

The default Earth Reference Model (ERM) used for geodetic positioning is 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 should set the *Earth Reference Model* parameter of the **Start of Frame** packet to Host-Defined (1). If, for some reason, the IG cannot support the ERM defined by the Host, the parameter should be set to WGS 84 (0).

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



*1 *Custom ERM Enable*

Figure 78 – Earth Reference Model Definition 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 – Earth Reference Model Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 19	This parameter identifies this data packet as the **Earth Reference Model Definition** packet. The value of this parameter must be 19.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.

Parameter	Description
Custom ERM Enable Type: 1-bit field Units: N/A Values: 0 Disable (use WGS 84) 1 Enable Default: 0	This parameter specifies whether the IG should use the Earth Reference Model (ERM) defined by this packet. If this parameter is set to Disable (0), the IG will use the WGS 84 reference model and all other parameters in this packet will be ignored.
Equatorial Radius Type: double float Units: meters Default: 6,378,137.0	This parameter specifies the semi-major axis of the ellipsoid.
Flattening Type: double float Units: meters Default: $\frac{1}{298.257223563}$	This parameter specifies the flattening of the ellipsoid. This value is 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.

4.1.20 Trajectory Definition

The **Trajectory Definition** packet enables the Host to describe a trajectory along which an IG-driven entity, such as a tracer round or particulate debris, travels. This is useful for simulating gravity and other static forces acting upon the entity. This packet is commonly used in conjunction with the **Rate Control** packet (Section 4.1.8).

The contents of the **Trajectory 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 ID = 20								*Packet Size* = 24								*Entity ID*															
Acceleration X																															
Acceleration Y																															
Acceleration Z																															
Retardation Rate																															
Terminal Velocity																															

Figure 79 – Trajectory 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 – Trajectory Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 20	This parameter identifies this data packet as the **Trajectory Definition** packet. The value of this parameter must be 20.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
Entity ID Type: unsigned int16 Units: N/A	This parameter identifies the entity for which the trajectory is defined.

Parameter	Description
Acceleration X Type: single float Units: m/s^2 Default: 0 Datum: Ellipsoid-tangential NED reference coordinate system	This parameter specifies the X component of the acceleration vector.
Acceleration Y Type: single float Units: m/s^2 Default: 0 Datum: Ellipsoid-tangential NED reference coordinate system	This parameter specifies the Y component of the acceleration vector.
Acceleration Z Type: single float Units: m/s^2 Default: 0 Datum: Ellipsoid-tangential NED reference coordinate system	This parameter specifies the Z component of the acceleration vector.
Retardation Rate Type: single float Units: m/s^2 Default: 0 Datum: Direction opposite of entity's instantaneous velocity vector	This parameter specifies the magnitude of an acceleration applied against the entity's instantaneous linear velocity vector. This is used to simulate drag and other frictional forces acting upon the entity.
Terminal Velocity Type: single float Units: m/s Default: 0	This parameter specifies the maximum velocity the entity can sustain.

4.1.21 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 will 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 will be used.

Refer to Section 3.2 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 ID = 21								*Packet Size* = 32								*View ID*															
Group ID								*7	*6	*5	*4	*3	*2	*1		*View Type*				*10	*9	*8		Reserved							
Near																															
Far																															
Left																															
Right																															
Top																															
Bottom																															

*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 80 – View 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 – View Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 21	This parameter identifies this data packet as the **View Definition** packet. The value of this parameter must be 21.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
View ID Type: unsigned int16 Units: N/A	This parameter specifies the view to which the data in this packet will be applied.
Group ID Type: unsigned int8 Units: N/A Values: 0 None 1 – 255 Specifies view group	This parameter specifies the group to which the view is to be assigned. If this value is zero (0), the view is not assigned to a group.
Near Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the near clipping plane will be set to the value of the *Near* parameter within this packet. If this parameter is set to Disable (0), the *Near* parameter will be ignored.
Far Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the far clipping plane will be set to the value of the *Far* parameter within this packet. If this parameter is set to Disable (0), the *Far* parameter will be ignored.
Left Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the left half-angle of the view frustum will be set according to the value of the *Left* parameter within this packet. If this parameter is set to Disable (0), the *Left* parameter will be ignored.

Parameter	Description
Right Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the right half-angle of the view frustum will be set according to the value of the *Right* parameter within this packet. If this parameter is set to Disable (0), the *Right* parameter will be ignored.
Top Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the top half-angle of the view frustum will be set according to the value of the *Top* parameter within this packet. If this parameter is set to Disable (0), the *Top* parameter will be ignored.
Bottom Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable	This parameter specifies whether the bottom half-angle of the view frustum will be set according to the value of the *Bottom* parameter within this packet. If this parameter is set to Disable (0), the *Bottom* parameter will 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	This parameter specifies 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	This parameter specifies 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	This parameter specifies whether the view projection should be perspective (Section 3.2.1.1) or orthographic parallel (Section 3.2.1.2).
Reorder Type: 1-bit field Units: N/A Values: 0 No Reorder 1 Bring to Top Default: IG-configurable	This parameter specifies whether the view should be moved to the top of any overlapping views. In cases where multiple overlapping views are moved to the top, the last view specified gets priority.
View Type Type: unsigned 3-bit field Units: N/A Values: 0 – 7 Default: IG-configurable	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.
Near Type: single float Units: meters Values: > 0 to < *Far* Default: IG-configurable	This parameter specifies the position of the view's near clipping plane. This distance is measured along the viewing vector from the eyepoint to the plane.
Far Type: single float Units: meters Values: > *Near* Default: IG-configurable	This parameter specifies the position of the view's far clipping plane. This distance is 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	This parameter specifies the left half-angle of the view frustum. This value is the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 19).
Right Type: single float Units: degrees Values: > *Left* to < 90.0 Default: IG-configurable	This parameter specifies the right half-angle of the view frustum. This value is the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 19).
Top Type: single float Units: degrees Values: > *Bottom* to < 90.0 Default: IG-configurable	This parameter specifies the top half-angle of the view frustum. This value is the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 19).
Bottom Type: single float Units: degrees Values: > -90.0 to < *Top* Default: IG-configurable	This parameter specifies the bottom half-angle of the view frustum. This value is the measure of the angle formed at the view eyepoint between the viewing vector and the frustum side (see Figure 13, page 19).

4.1.22 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 along which collision testing is performed by the IG. When a collision detection segment intersects a polygon, the IG registers a collision by sending a **Collision Detection Segment Notification** (Section 4.2.13) packet to the Host identifying the segment and the object with which it collided.

Note that collision detection testing is 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 81 illustrates five segments defined for an aircraft:

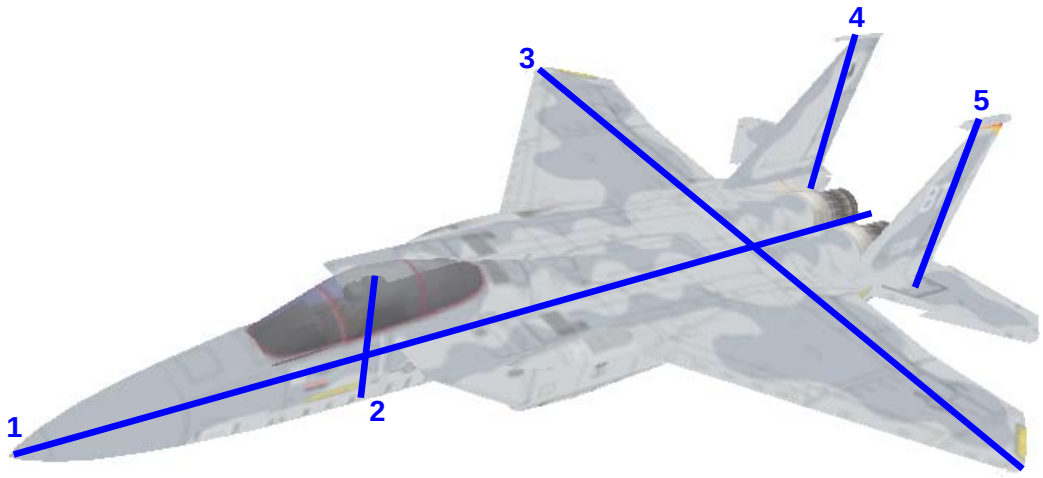


Figure 81 – Examples of Collision Detection Segments

Collision detection volumes are tested segment-to-polygon. An entity will not perform collision detection segment testing against its own geometry.

If the *Collision Detection Enable* parameter of an **Entity Control** packet is set to Disabled (0), the referenced entity's segments will 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 will be included in collision detection segment testing.

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

Although non-entity collision detection segments may be defined by the IG configuration, the Host can only create collision detection segments by referencing an entity. If a segment must be defined along a non-entity object, the Host must 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 ID = 22								*Packet Size* = 40								*Entity ID*															
Segment ID								Reserved							*1	Reserved															
X1																															
Y1																															
Z1																															
X2																															
Y2																															
Z2																															
Material Mask																															
Reserved																															

*1 *Segment Enable*

Figure 82 – Collision Detection Segment 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 Segment Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 22	This parameter identifies this data packet as the **Collision Detection Segment Definition** packet. The value of this parameter must be 22.
Packet Size Type: unsigned int8 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 40.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity for which the segment is defined.
Segment ID Type: unsigned int8 Units: N/A	This parameter specifies the ID of the segment. If an ID is specified for which a segment is already defined, that segment will be overwritten.

Parameter	Description
Segment Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the segment is enabled or disabled. If it is set to Disable (0), the specified segment is ignored during collision testing.
X1 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies the X offset of one endpoint of the collision segment. This offset is 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	This parameter specifies the Y offset of one endpoint of the collision segment. This offset is 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	This parameter specifies the Z offset of one endpoint of the collision segment. This offset is 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	This parameter specifies the X offset of one endpoint of the collision segment. This offset is 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.

Parameter	Description
Y2 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies the Y offset of one endpoint of the collision segment. This offset is 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.
Z2 Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies the Z offset of one endpoint of the collision segment. This offset is 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	This parameter specifies 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) will cause the IG to register hits with materials within the corresponding range. Refer to the appropriate IG documentation for material code assignments.

4.1.23 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 registers a collision by sending a **Collision Detection Volume Notification** (Section 4.2.14) packet to the Host identifying the collided volumes.

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

A volume is defined 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 83 illustrates two cuboid volumes defined for an aircraft:

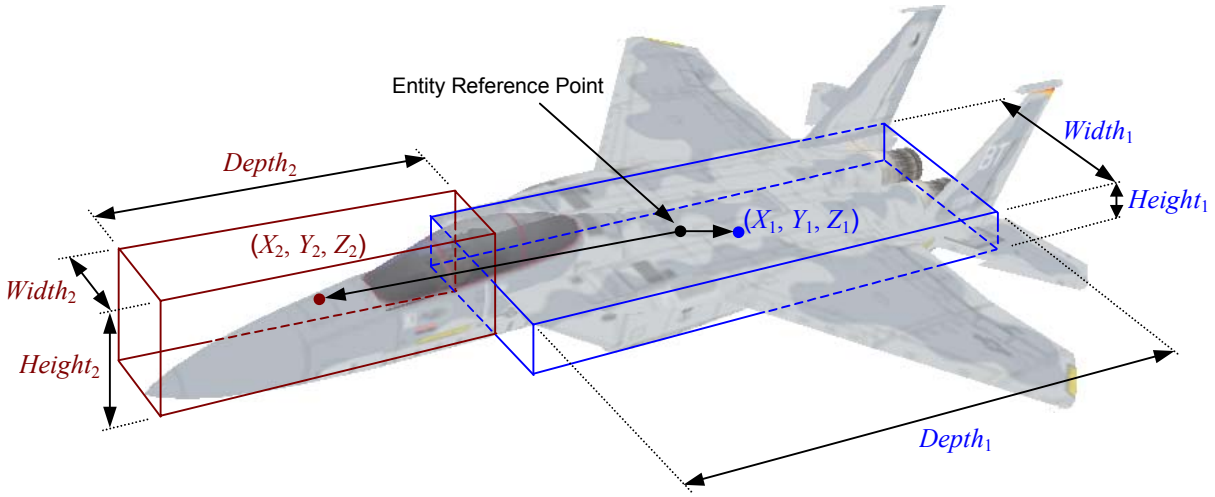


Figure 83 – Examples of Collision Detection Volumes

Unlike collision detection segments, which are tested segment-to-polygon, collision detection volumes are tested volume-to-volume. Volumes associated with the same entity are not 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, no collision detection volume testing will be performed for that entity.

If the *Collision Detection Enable* parameter of the **Entity Control** packet is set to Disabled (0), no volumes defined for the entity will be used as "source" volumes for collision testing. Figure 84 below illustrates the individual tests that would occur each frame between a hypothetical group of entities:

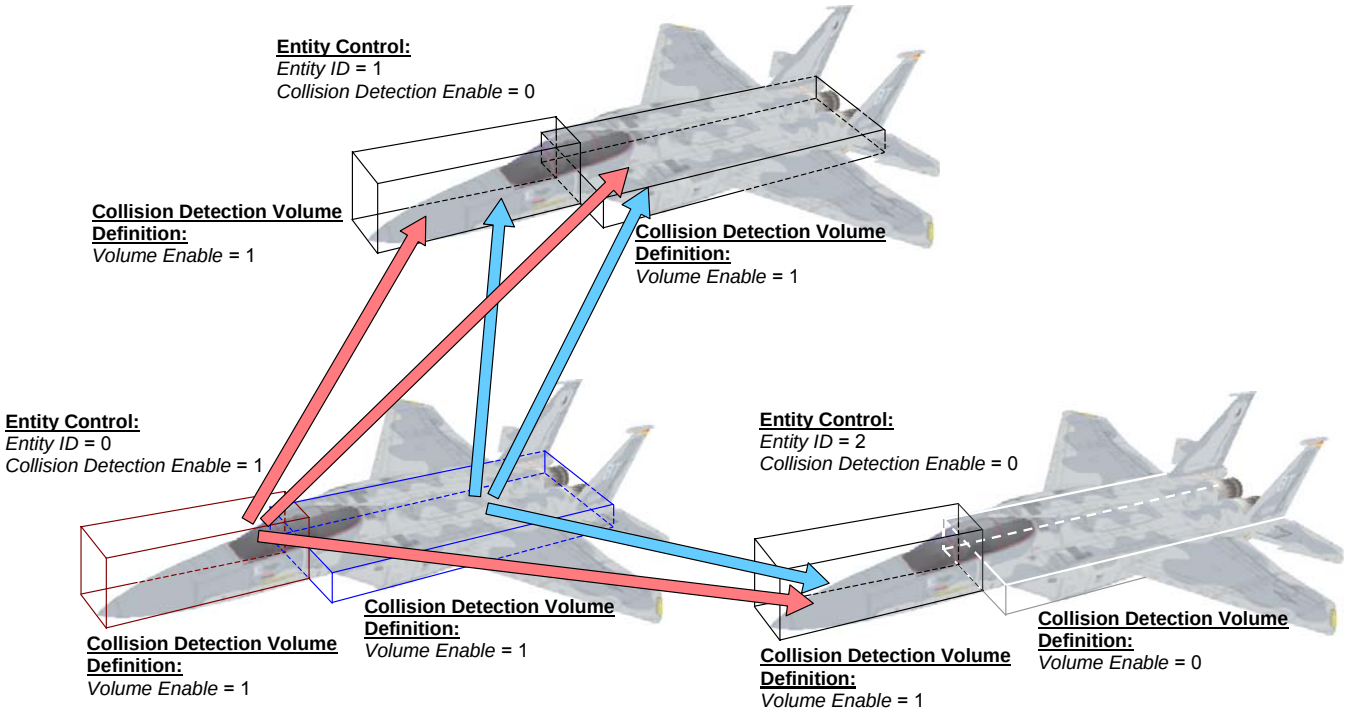


Figure 84 – Collision Volume Testing Between Multiple Entities

The illustration shows three aircraft with two collision detection volumes defined for each. The *Collision Detection Enable* parameter has been set to Enabled (1) for the Ownship (Entity 0) and to Disabled (0) for Entities 1 and 2. This means that only the volumes associated with the Ownship will be used as the sources of collision testing. The two source volumes are tested with every other enabled volume not associated with the Ownship. 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 entities, two tests will be performed between each pair of volumes. This is because each volume will be used as both source and destination in each pair-wise test.

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

Although non-entity collision detection volumes may be defined by the IG configuration, the Host can only create collision detection volumes by referencing an entity. If a volume must be defined about a non-entity object, the Host must 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 ID = 23								*Packet Size* = 48								*Entity ID*															
Volume ID								Reserved						*2	*1	Reserved															
X																															
Y																															
Z																															
Height/Radius																															
Width																															
Depth																															
Roll																															
Pitch																															
Yaw																															
Reserved																															

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

Figure 85 – Collision Detection Volume Definition 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 – Collision Detection Volume Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 23	This parameter identifies this data packet as the **Collision Detection Volume Definition** packet. The value of this parameter must be 23.
Packet Size Type: unsigned int8 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 48.
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity for which the volume is defined.
Volume ID Type: unsigned int8 Units: N/A	This parameter specifies the ID of the volume. If an ID is specified for which a volume is already defined, that volume will be overwritten.

Parameter	Description
Volume Enable Type: 1-bit field Units: N/A Values: 0 Disable 1 Enable Default: IG-configurable	This parameter specifies whether the volume is enabled or disabled. If it is set to Disable (0), the specified volume is ignored during collision testing.
Volume Type Type: 1-bit field Units: N/A Values: 0 Sphere 1 Cuboid Default: IG-configurable	This parameter specified whether the volume is spherical or cuboid.
X Type: single float Units: meters Default: IG-configurable Datum: Entity reference point	This parameter specifies the X offset of the center of the volume. This offset is 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	This parameter specifies the Y offset of the center of the volume. This offset is 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	This parameter specifies the Z offset of the center of the volume. This offset is 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, this parameter specifies the radius of the sphere.
Height (Cuboid Volumes) Type: single float Units: meters Values: > 0 Default: IG-configurable	For cuboid collision detection volumes, this parameter specifies 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, this parameter specifies the length of the cuboid along its **Y** axis. This parameter is 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, this parameter specifies the length of the cuboid along its **X** axis. This parameter is 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, this parameter specifies the roll of the cuboid with respect to the entity's coordinate system. This parameter is 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, this parameter specifies the pitch of the cuboid with respect to the entity's coordinate system. This parameter is 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, this parameter specifies the yaw of the cuboid with respect to the entity's coordinate system. This parameter is ignored if *Volume Type* is set to Sphere (0).

4.1.24 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 will set the HAT/HOT *ID* parameter of the response packet to match that in the request.

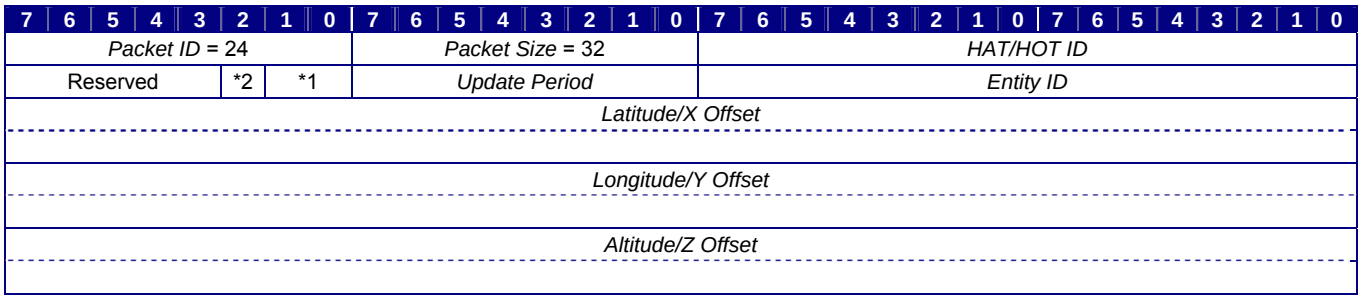
The *Update Period* parameter specifies the number of frames between periodic responses. This allows the Host to send just one **HAT/HOT Request** packet but receive continuous responses if the test point will not move with respect to the specified coordinate system. If *Update Period* is set to zero,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

If the *Request Type* parameter is set to HAT (0) or HOT (1), the IG will respond with a **HAT/HOT Response** packet (Section 4.2.2) containing the requested datum. If the parameter is set to Extended HAT/HOT (2), the IG will respond with a **HAT/HOT Extended Response** packet (Section 4.2.3) 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 will 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 can 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:



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

Figure 86 – HAT/HOT 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 – HAT/HOT Request Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 24	This parameter identifies this data packet as the **HAT/HOT Request** packet. The value of this parameter must be 24.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
HAT/HOT ID Type: unsigned int16 Units: N/A	This parameter identifies the HAT/HOT request. 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 will contain this value to correlate the response with this request.
Request Type Type: unsigned 2-bit field Units: N/A Values: 0 HAT 1 HOT 2 Extended	This parameter determines what type of response packet the IG should return for this request. If this parameter is set to HAT (0), the IG will respond with a **HAT/HOT Response** packet containing the Height Above Terrain. If this parameter is set to HOT (1), the IG will respond with a **HAT/HOT Response** packet containing the Height Of Terrain. If this parameter is set to Extended (2), the IG will respond with a **HAT/HOT Extended Response** packet, which contains both the Height Above Terrain and the Height Of Terrain.
Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	This parameter specifies the coordinate system within which the test point is defined. If this parameter is set to Geodetic (0), the test point is defined as a Latitude, Longitude, and Altitude. If this parameter is set to Entity (1), the test point is 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	This parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG should return a single response. A value of n > 0 indicates that the IG should return a response every n^{th} frame.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity relative to which the test point is defined. This parameter is ignored if **Coordinate System** is set to Geodetic (0).
Latitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -90.0 – 90.0 Datum: Equator	This parameter specifies the latitude from which the HAT/HOT request is being made.
X Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	This parameter specifies the X offset of the point from which the HAT/HOT request is being made. This value is given relative to the entity's reference point.
Longitude (Geodetic Coordinate System) Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	This parameter specifies the longitude from which the HAT/HOT request is being made.
Y Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	This parameter specifies the Y offset of the point from which the HAT/HOT request is being made. This value is given relative to the entity's reference point.
Altitude (Geodetic Coordinate System) Type: double float Units: meters Datum: Mean Sea Level	This parameter specifies the altitude from which the HAT/HOT request is being made. This parameter is ignored if *Request Type* is set to HOT (1).
Z Offset (Entity Coordinate System) Type: double float Units: meters Datum: Entity Reference Point	This parameter specifies the Z offset of the point from which the HAT/HOT request is being made. This value is given relative to the entity's reference point.

4.1.25 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 will 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. Duplicating the *LOS ID* value between both request types can cause data loss.

If the *Request Type* parameter is set to Basic (0), the IG will respond with a **Line of Sight Response** packet (Section 4.2.4). If the parameter is set to Extended (1), the IG will respond with a **Line of Sight Extended Response** packet (Section 4.2.5).

The *Alpha Threshold* parameter specifies the minimum alpha value with which an intersection should register. 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 will 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 will not move with respect to the specified coordinate system. If *Update Period* is set to zero, the request will be treated as a one-shot request and the IG will return a single response. The Host should manipulate the value of *LOS ID* so that an ID is not reused before the IG has sufficient time to process and respond to the request. If *Update Period* is set to some value n greater than zero, the IG will 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 will be set to zero (0).

The IG will 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 can 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 ID = 25								*Packet Size* = 64								*LOS ID*															
Reserved			*5	*4	*3	*2	*1	*Alpha Threshold*								*Source Entity ID*															
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																															
Update Period								Reserved								*Destination Entity ID*															

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

Figure 87 – Line of Sight Segment 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 Segment Request Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 25	This parameter identifies this data packet as the **Line of Sight Segment Request** packet. The value of this parameter must be 25.
Packet Size Type: unsigned int8 Units: Bytes Value: 64	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 64.

Parameter	Description
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS request. When the IG returns a **Line of Sight Response** packet in response to this request, the *LOS ID* parameter of that packet will 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.
Request Type Type: 1-bit field Units: N/A Values: 0 Basic 1 Extended	This parameter determines what type of response the IG should return for this request. If this parameter is set to Basic (0), the IG will respond with a **Line of Sight Response** packet. If this parameter is set to Extended (1), the IG will 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	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 is given by latitude, longitude, and altitude. If this parameter is set to Entity (1), then the endpoint is defined relative to the reference point of the entity specified by *Entity ID*.
Destination Point Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	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 is 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 is 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 is 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	This parameter specifies the coordinate system to be used in the response. If this parameter is set to Geodetic (0), then the intersection point will be specified by latitude, longitude, and altitude. If this parameter is set to Entity (1), then the intersection point will be specified relative to the reference point of the intersected entity.

Parameter	Description
Destination Entity ID Valid Type: unsigned int16 Units: N/A Values: 0 Not Valid 1 Valid	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 will 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 will 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	This parameter specifies the minimum alpha value (i.e., minimum opacity) a surface may have for an LOS response to be generated.
Source Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity relative to which the test segment endpoints are defined. This parameter is ignored if *Source Point Coordinate System* and *Destination Point Coordinate System* are both set to Geodetic (0).
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 specifies 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 specifies the X offset of the source endpoint of the LOS test segment.

Parameter	Description
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 specifies 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 specifies the Y offset of the source endpoint of the LOS test segment.
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 specifies 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 specifies 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 specifies 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 specifies 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.

Parameter	Description
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 specifies 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 specifies 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.
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 specifies 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 specifies 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) will 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.
Update Period Type: unsigned int8 Units: N/A Values: 0 One-Shot request > 0 Indicates update period	This parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG should return a single response. A value of $n > 0$ indicates that the IG should return a response every n^{th} frame.

Parameter	Description
Destination Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity with respect to which the *Destination X Offset*, *Destination Y Offset*, and *Destination Z Offset* parameters are specified. This parameter is 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).

4.1.26 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 a LOS request, it will copy the *LOS ID* value contained within the request to the *LOS ID* parameter of the corresponding response packet. 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. This will prevent similarly identified requests from being lost by the IG.

Note that **Line of Sight Segment Request** packets and **Line of Sight Vector Request** packets share the *LOS ID* parameter. Duplicating the *LOS ID* value between both request types can also cause data loss.

If the *Request Type* parameter is set to Basic (0), the IG will respond with a **Line of Sight Response** packet (Section 4.2.4). If the parameter is set to Extended (1), the IG will respond with a **Line of Sight Extended Response** packet (Section 4.2.5).

The *Alpha Threshold* parameter specifies the minimum alpha value with which an intersection should register. 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 will 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 will not move with respect to the specified coordinate system. If *Update Period* is set to zero, the request will be treated as a one-shot request and the IG will return a single response. The Host should manipulate the value of *LOS ID* so that an ID is not reused before the IG has sufficient time to process and respond to the request. If *Update Period* is set to some value n greater than zero, the IG will 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 will be set to zero (0).

The IG will 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 can 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 ID = 26								*Packet Size* = 56								*LOS ID*															
Reserved					*3	*2	*1	*Alpha Threshold*								*Entity ID*															
Azimuth																															
Elevation																															
Minimum Range																															
Maximum Range																															
Source Latitude/X Offset																															
Source Longitude/Y Offset																															
Source Altitude/Z Offset																															
Material Mask																															
Update Period								Reserved																							

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

Figure 88 – Line of Sight Vector 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 – Line of Sight Vector Request Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 26	This parameter identifies this data packet as the **Line of Sight Vector Request** packet. The value of this parameter must be 26.
Packet Size Type: unsigned int8 Units: Bytes Value: 56	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 56.

Parameter	Description
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS request. When the IG returns a **Line of Sight Response** packet in response to this request, the *LOS ID* parameter of that packet will 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.
Request Type Type: 1-bit field Units: N/A Values: 0 Basic 1 Extended	This parameter determines what type of response the IG should return for this request. If this parameter is set to Basic (0), the IG will respond with a **Line of Sight Response** packet. If this parameter is set to Extended (1), the IG will 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	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 is given by latitude, longitude, and altitude. The vector, specified by *Azimuth* and *Elevation*, is defined relative to the Geodetic coordinate system. If this parameter is set to Entity (1), then the point is defined relative to the reference point of the entity specified by *Entity ID*. The vector is also specified relative to the entity's coordinate system. Note: The value of this parameter also determines the coordinate system in which the response data are given.
Response Coordinate System Type: 1-bit field Units: N/A Values: 0 Geodetic 1 Entity	This parameter specifies the coordinate system to be used in the response. If this parameter is set to Geodetic (0), then the intersection point will be specified by latitude, longitude, and altitude. If this parameter is set to Entity (1), then the intersection point will be specified relative to the reference point of the intersected entity.
Alpha Threshold Type: unsigned int8 Units: N/A	This parameter specifies the minimum alpha value (i.e., minimum opacity) a surface may have for an LOS response to be generated.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter specifies the entity relative to which the test segment endpoints are defined. This parameter is ignored if *Source Point Coordinate System* is set to Geodetic (0).
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	This parameter specifies 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	This parameter specifies the vertical angle of the LOS test vector. For vectors defined relative to the geodetic reference plane, a positive elevation produces a vector away from the ellipsoid. For vectors defined relative to an entity, a positive elevation produces an acute angle with the entity's –**Z** (up) axis.
Minimum Range Type: single float Units: meters Values: ≥ 0 Datum: LOS test vector source point	This parameter specifies the minimum range along the LOS test vector at which intersection testing should occur.
Maximum Range Type: single float Units: meters Values: > *Minimum Range* Datum: LOS test vector source point	This parameter specifies the maximum range along the LOS test vector at which intersection testing should occur.

Parameter	Description
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 specifies 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 specifies 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 specifies 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 specifies the Y offset of the source point of the LOS test vector.
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 specifies 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 specifies the Z offset of the source point of the LOS test vector.

Parameter	Description
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) will 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.
Update Period Type: unsigned int8 Units: N/A Values: 0 One-Shot request > 0 Indicates update period	This parameter specifies the interval between successive responses to this request. A value of zero (0) indicates that the IG should return a single response. A value of n > 0 indicates that the IG should return a response every n^{th} frame.

4.1.27 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 is useful for determining the locations of autonomous IG-driven entities, child entities and articulated parts, and view eyepoints. It can also be used for determining the instantaneous position and orientation of head trackers and other tracked input devices.

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 ID = 27								*Packet Size* = 8								*Object ID*															
Articulated Part ID								*4				*3				*2								*1				Reserved			

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

Figure 89 – Position 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 – Position Request Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 27	This parameter identifies this data packet as the **Position Request** packet. The value of this parameter must be 27.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.
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 *Object Class* is set to Articulated Part (1), this parameter specifies the entity whose part is identified by the *Articulated Part ID* parameter.

Parameter	Description
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 is specified by the *Object ID* parameter. This parameter is valid only when *Object Class* is set to Articulated Part (1).
Update Mode Type: 1-bit field Units: N/A Values: 0 One-Shot 1 Continuous	This parameter specifies whether the IG should report the position of the requested object each frame. If this parameter is set to One-Shot (0), the IG should report the position only one time.
Object Class Type: unsigned 3-bit field Units: N/A Values: 0 Entity 1 Articulated Part 2 View 3 View Group 4 Motion Tracker	This parameter specifies 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	This parameter specifies the desired coordinate system relative to which the position and orientation should be given. **Geodetic** – Position will be specified as a geodetic latitude, longitude, and altitude. Orientation is given with respect to the reference plane shown in Figure 18, page 25. **Parent Entity** – Position and orientation are with respect to the entity to which the specified entity or view is attached. This value is invalid for top-level entities. **Submodel** – Position and orientation will be specified with respect to the articulated part's reference coordinate system as described in Section 3.4.3. This value is valid only when *Object Class* is set to Articulated Part (1). Note: If *Object Class* is set to Motion Tracker (3), The coordinate system is defined by the tracking device and this parameter is ignored.

4.1.28 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.

For a given test point, the IG may respond with no more than one of each of the **Maritime Surface Conditions Response** and **Weather Conditions Response** packets. For terrestrial surface conditions requests, the IG should 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 should be returned, the IG must 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 90:

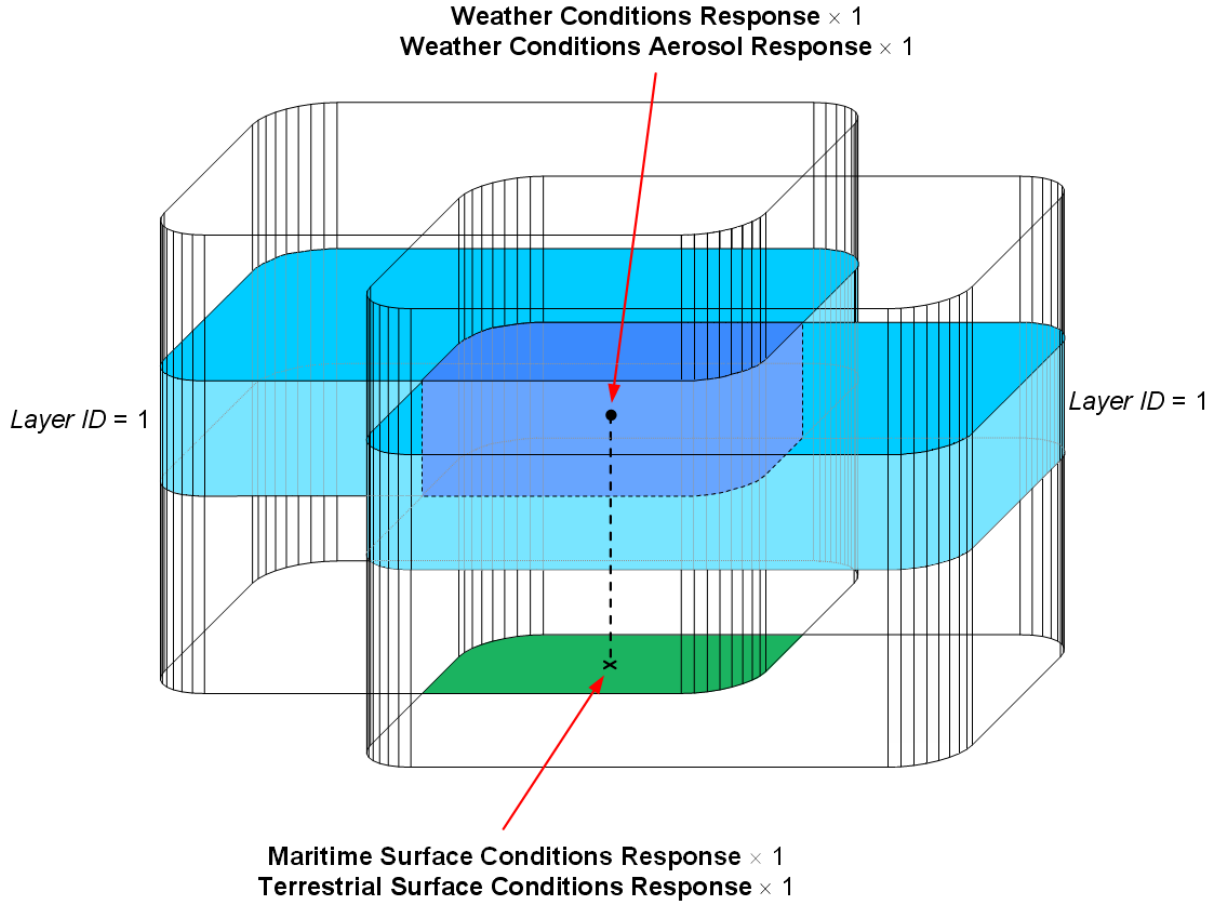


Figure 90 – Environmental Conditions Request (Weather Layers with Same ID)

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 91.

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. The following diagram illustrates this exchange of data between the Host and IG:

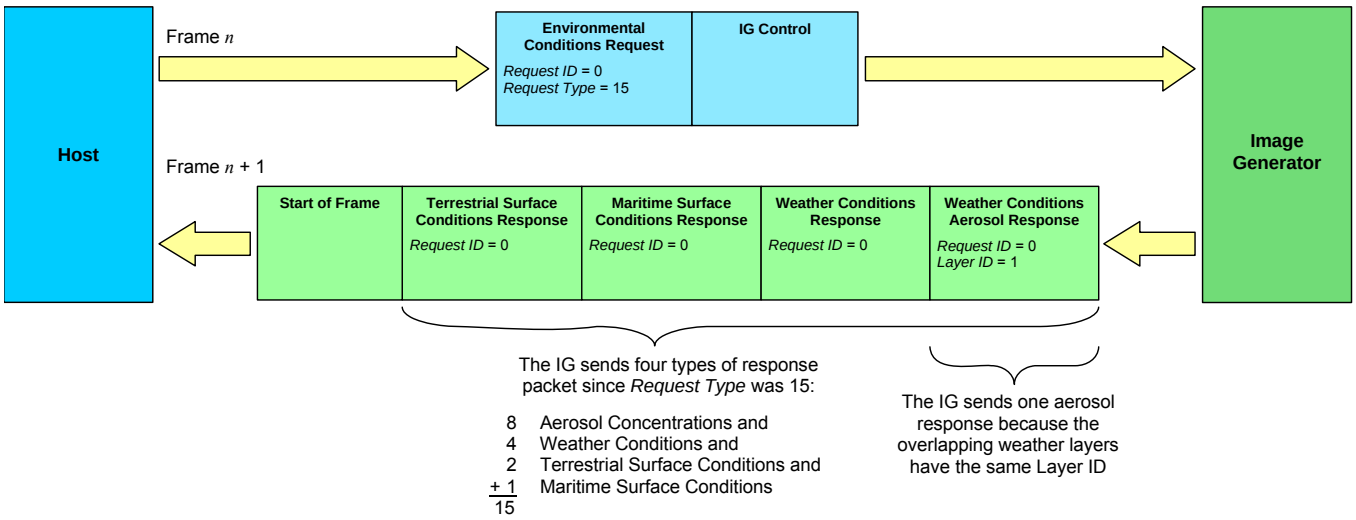


Figure 91 – Data Exchange for Environmental Conditions Request (One Aerosol)

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.

On the other hand, the overlapping weather layers might have different layer IDs as shown below:

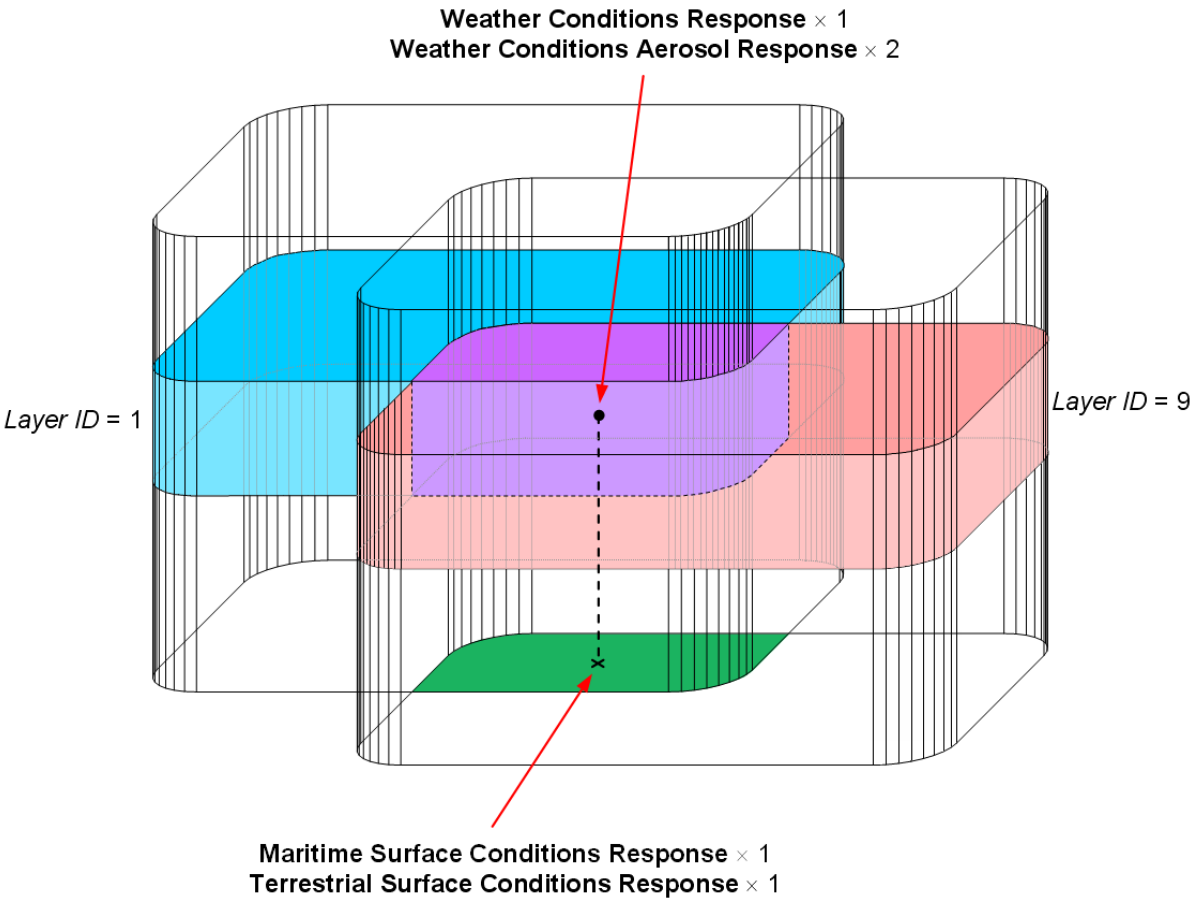


Figure 92 – Environmental Conditions Request (Weather Layers with Different IDs)

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:

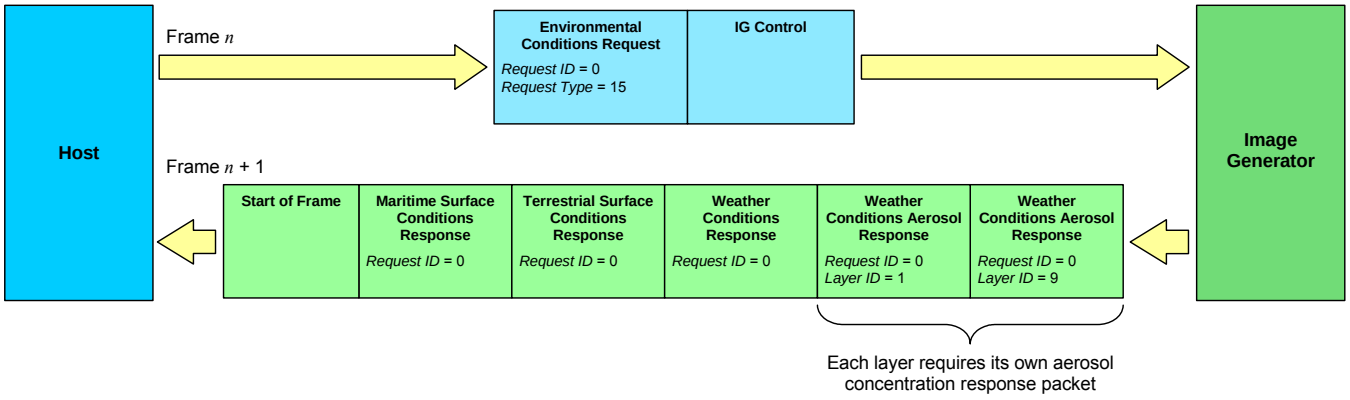


Figure 93 – Data Exchange for Environmental Conditions Request (Two Aerosols)

The *Layer ID* parameter of each response packet would correspond 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 ID = 28								*Packet Size* = 32								Reserved								*1								*Request ID*							
Reserved																															
Latitude																															
Longitude																															
Altitude																															

*1 *Request Type*

Figure 94 – Environmental Conditions Request 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 – Environmental Conditions Request Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 28	This parameter identifies this data packet as the **Environmental Conditions Request** packet. The value of this parameter must be 28.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.

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 will respond with a **Maritime Surface Conditions Response** packet (Section 4.2.11). **Terrestrial Surface Conditions** – The IG will respond with a **Terrestrial Surface Conditions Response** packet (Section 4.2.12). **Weather Conditions** – The IG will respond with a **Weather Conditions Response** packet (Section 4.2.9). **Aerosol Concentrations** – The IG will send exactly one **Aerosol Concentration Response** packet (Section 4.2.10) 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. When the IG returns a responds to the request, each response packet(s) will 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.
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.
Altitude Type: double float Units: meters Datum: Mean Sea Level	This parameter specifies the geodetic altitude at which the environmental state is requested. This parameter is used only for weather conditions and aerosol concentrations requests.

4.1.29 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 is 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 creates a new surface based on the packet's parameters. If the surface does exist, it is modified according to the packet's parameters.

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

For surfaces attached to a view, the *Left*, *Right*, *Top*, and *Bottom* parameters define the position and size of the surface. These values are specified relative to the view's Normalized Viewport Coordinate System as described in Section 3.4.4.3.

For non-billboard 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 is the same as for a child entity and is described in Section 3.4.4.1. The *Width* and *Height* parameters specify the size of the surface.

For billboard surfaces attached to an entity, the *X Offset*, *Y Offset*, and *Z Offset* parameters define the distance from the surface to the entity. The orientation of the entity has no effect on the view's orientation; the view is always parallel to the view plane as described in Section 3.4.4.2. The *Width* and *Height* parameters specify the size of the surface. The *Yaw*, *Pitch*, and *Roll* parameters are ignored.

Every surface has a local 2D (UV) coordinate system that is used to place, rotate, and size of each symbol drawn on the surface as described in Section 3.4.5.1. 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 is 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 is drawn first. Surface 4 is drawn next and may occult any overlapping areas. Finally, Surface 7 is drawn on top and may likewise occult parts of the other surfaces.

Any surface attached to an entity is contained in the scene and is drawn with entities and other objects also in the scene. Since surfaces attached to views are coincident with the near clipping plane, view-attached surfaces are drawn on top of all other objects in the view.

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

A symbol surface is destroyed by setting the *Surface State* parameter to Destroyed (1). Any symbols associated with that surface are also destroyed.

If an entity is destroyed, then any symbol surfaces attached to that entity are also 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 ID = 29								*Packet Size* = 56								*Surface ID*															
Reserved				*4	*3	*2	*1	Reserved								*Entity ID/View ID*															
X Offset/Left																															
Y Offset/Right																															
Z Offset/Top																															
Yaw/Bottom																															
Pitch																															
Roll																															
Width																															
Height																															
Min U																															
Max U																															
Min V																															
Max V																															

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

Figure 95 – Symbol Surface 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 Surface Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 29	This parameter identifies this data packet as the **Symbol Surface Definition** packet. The value of this parameter must be 29.
Packet Size Type: unsigned int8 Units: Bytes Value: 56	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 56.
Surface ID Type: unsigned int16 Units: N/A Values: N/A	This parameter specifies the symbol surface to which this packet is applied.

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

Parameter	Description
Perspective Growth Enable Type: 1-bit field Units: N/A Values: 0 Disabled 1 Enabled	This parameter specifies whether the surface appears to maintain a constant size or has perspective growth as the entity to which the surface is attached moves closer to 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 will appear to stay the same size (i.e., will 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 will 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 this parameter is ignored. If the surface is attached to a view, this parameter is ignored.
Entity ID/View ID Type: unsigned int16 Units: N/A	This parameter specifies the entity or view to which this surface is attached.
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, this parameter specifies the distance along the entity's **X** axis from the entity's reference point to the center of the surface (see Section 3.4.4.1). For a billboard surface attached to an entity, this parameter specifies the distance along the surface's **X** axis from the center of the surface to the entity's reference point (see Section 3.4.4.2).
Left (View-attached surfaces) Type: single float Units: viewport width Datum: Normalized viewport coordinate system	For a surface attached to a view, this parameter specifies the distance from the left edge of the viewport to the surface's leftmost boundary as a fraction of the viewport's width (see Section 3.4.4.3).

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, this parameter specifies the distance along the entity's **Y** axis from the entity's reference point to the center of the surface (see Section 3.4.4.1). For a billboard surface attached to an entity, this parameter specifies the distance along the surface's **Y** axis from the center of the surface to the entity's reference point (see Section 3.4.4.2).
Right (View-attached surfaces) Type: single float Units: viewport width Datum: Normalized viewport coordinate system	For a surface attached to a view, this parameter specifies the distance from the left edge of the viewport to the surface's rightmost boundary as a fraction of the viewport's width (see Section 3.4.4.3).
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, this parameter specifies the distance along the entity's **Z** axis from the entity's reference point to the center of the surface (see Section 3.4.4.1). For a billboard surface attached to an entity, this parameter specifies the distance along the surface's **Z** axis from the center of the surface to the entity's reference point (see Section 3.4.4.2).
Top (View-attached surfaces) Type: single float Units: viewport height Datum: Normalized viewport coordinate system	For a surface attached to a view, this parameter specifies the distance from the bottom edge of the viewport to the surface's topmost boundary as a fraction of the viewport's height (see Section 3.4.4.3).
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, this parameter specifies a rotation about the surface's **Z** axis as described in Section 3.4.4.1 For entity-attached billboard surfaces, this parameter is ignored.
Bottom (View-attached surfaces) Type: single float Units: viewport width Datum: Normalized viewport coordinate system	For a surface attached to a view, this parameter specifies the distance from the bottom edge of the viewport to the surface's bottommost boundary as a fraction of the viewport's height (see Section 3.4.4.3).

Parameter	Description
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, this parameter specifies a rotation about the surface's **Y** axis as described in Section 3.4.4.1 For entity-attached billboard surfaces, this parameter is ignored. For a surface attached to a view, this parameter is 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, this parameter specifies a rotation about the surface's **X** axis as described in Section 3.4.4.1 For entity-attached billboard surfaces, this parameter is ignored. For a surface attached to a view, this parameter is ignored.
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 this parameter specifies the width of the surface in meters. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Enabled (1), then this parameter specifies the width of the surface in meters. The apparent size of the surface will 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 this parameter specifies the width as a view arc and the occupied view space remains constant regardless of the surface's distance from the eyepoint. If the surface is attached to a view, this parameter is ignored.

Parameter	Description
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 this parameter specifies the width of the surface in meters. If the surface is attached to an entity and is a billboard, and if *Perspective Growth Enable* is set to Enabled (1), then this parameter specifies the width of the surface in meters. The apparent size of the surface will 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 this parameter specifies the width as a view arc and the occupied view space remains constant regardless of the surface's distance from the eyepoint. If the surface is attached to a view, this parameter is ignored.
Min U Type: single float Units: Symbol surface horizontal units Values: < *Max U*	This parameter specifies the minimum U coordinate of the symbol surface's viewable area. In other words, this parameter specifies the U coordinate that will correspond to the leftmost boundary of the symbol surface. Symbol surface 2D coordinate systems and horizontal units are described in Section 3.4.5.1.
Max U Type: single float Units: Symbol surface horizontal units Values: > *Min U*	This parameter specifies the maximum U coordinate of the symbol surface's viewable area. In other words, this parameter specifies the U coordinate that will correspond to the rightmost boundary of the symbol surface. Symbol surface 2D coordinate systems and horizontal units are described in Section 3.4.5.1.
Min V Type: single float Units: Symbol surface vertical units Values: > *Max V*	This parameter specifies the minimum V coordinate of the symbol surface's viewable area. In other words, this parameter specifies the U coordinate that will correspond to the bottommost boundary of the symbol surface. Symbol surface 2D coordinate systems and vertical units are described in Section 3.4.5.1.
Max V Type: single float Units: Symbol surface vertical units Values: < *Min V*	This parameter specifies the maximum V coordinate of the symbol surface's viewable area. In other words, this parameter specifies the U coordinate that will correspond to the topmost boundary of the symbol surface. Symbol surface 2D coordinate systems and vertical units are described in Section 3.4.5.1.

4.1.30 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 must 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 Line Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing text symbol will be destroyed along with any children and a new symbol will 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 shall be identified using a separate *Font ID*. Table 37 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 is composed of multiple UTF-8 character data. The text must 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 must be padded with zeroes up to the next eight-byte boundary. Zero-length text strings must 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 and, therefore, to a large extent, the language being used.

The *Packet Size* parameter must 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 must 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 96 illustrates the byte allocation for the packet:

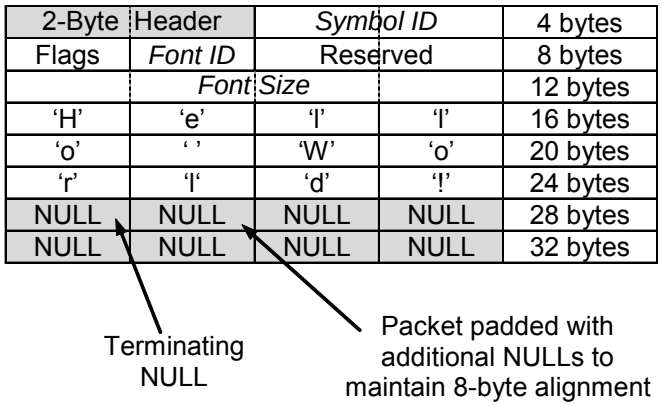
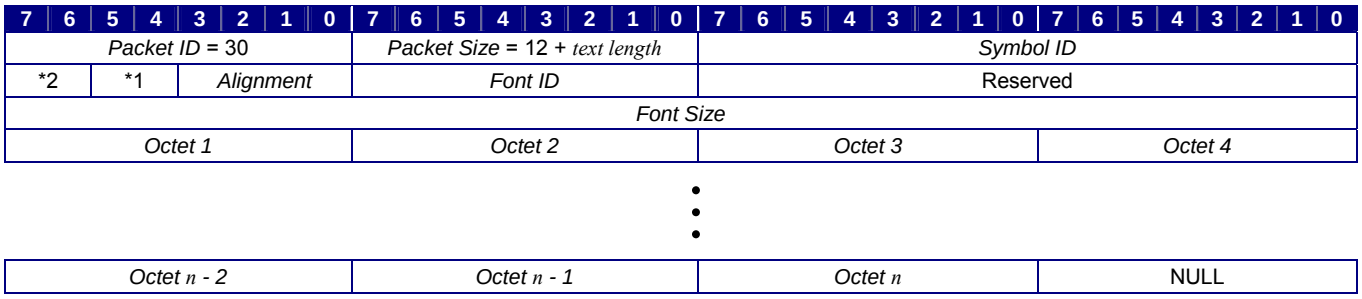


Figure 96 – Example of Symbol Text Definition Packet

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

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



*1 *Orientation*
*2 Reserved

Figure 97 – Symbol Text 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 Text Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 30	This parameter identifies this data packet as the **Symbol Text Definition** packet. The value of this parameter must be 30.
Packet Size Type: unsigned int8 Units: bytes Value: 16 ≤ (12 + *text length*) ≤ 248	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 12 plus the length of the text, including NULL characters. The minimum size of the **Symbol Text Definition** packet is 16 bytes. The maximum packet size is 248 bytes. This allows for a message length of up to 236 octets, including the terminating NULL. Note: Because all packets must begin and end on a 64-bit boundary, the value of this parameter must be an even multiple of eight (8).
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier must 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 that symbol and any children will be destroyed and a new symbol created.

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 of the symbol's reference point in relation to the text. If the text has multiple lines, this parameter also determines whether the text is 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. 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 Proportional: 1 Sans Serif 2 **Serif Bold** 3 Sans Serif *Italic* 4 ***Sans Serif Bold Italic*** 5 Serif 6 **Serif Bold** 7 Serif *Italic* 8 ***Serif Bold Italic*** Monospace: 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 ***Bold Italic*** 17–255 IG-defined	This parameter specifies the font to be used for this text symbol. This document defines a set of default proportional (variable-width) and monospace (constant-width) font styles for interoperability; however, the exact typefaces used will be IG-dependent. Font IDs 17 through 255 are entirely IG-defined.
Font Size Type: single float Units: Symbol surface vertical units	This parameter specifies the font size in terms of the vertical units defined by the symbol surface's 2D coordinate system (see Section 3.4.5.1).
Octet n Type: octet Units: N/A	These 8-bit data are used to store the UTF-8 code points in the string. Note: The maximum length of the string, including a terminating NULL, is 248 bytes.

4.1.31 Symbol Circle Definition

The **Symbol Circle Definition** packet is used to create a single circle or arc. This packet can also be used to create a composite symbol composed of up to 10 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 must 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 Line Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle symbol will be destroyed along with any children and a new symbol will be created using the new definition packet.

The center of each circle is located at a point (u, v) on the symbol's 2D coordinate system (see Section 3.4.5.2) as defined by the *Center U* and *Center V* parameters. The radius of the circle is 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 will appear as an ellipse and an arc will appear as an elliptical arc.

The *Start Angle* and *End Angle* parameters define the endpoints of the curve. These angles are measured counter-clockwise from the symbol's +**U** axis as shown below in Figure 98 and Figure 99. If these two values are equal, then the symbol defines a full circle. If these two values are not equal, then the symbol defines an arc. For circles, it is recommended that values of 0.0 be used for consistency and to avoid floating-point errors.

A circle can either be drawn as a curved line along the circumference or be filled, depending upon the value of the *Drawing Style* parameter. If *Drawing Style* is set to Line (0), then a curve is drawn from the start angle to the end angle with the specified *Radius*. The *Inner Radius* parameter is ignored.

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 will 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 will be drawn; if the bit is cleared (0) then the corresponding section will not be drawn. If the value of this parameter is 0xFFFF, then the line is solid.

The length of each section is 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 is repeated.

Figure 98 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.

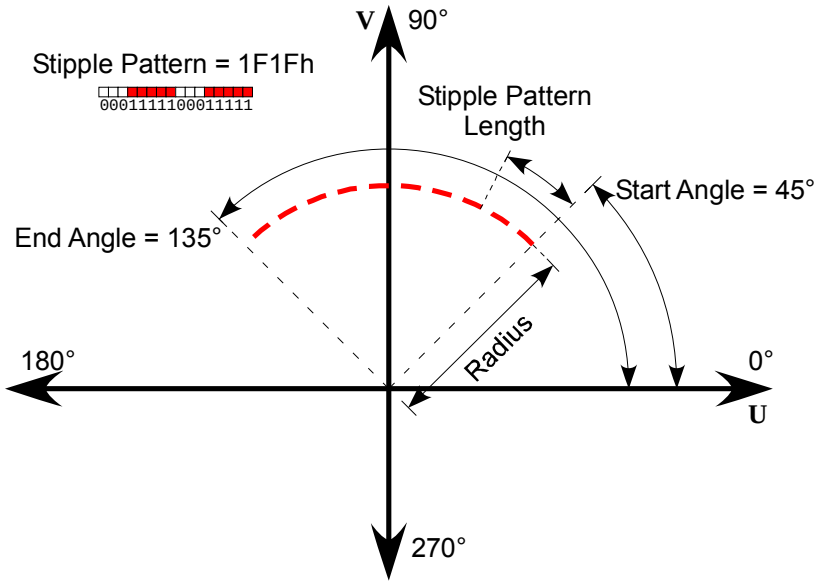


Figure 98 – Example of Line Circle (Arc) Symbol

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 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 circle is completely filled.

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

Figure 99 shows an example of a partially filled arc 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.

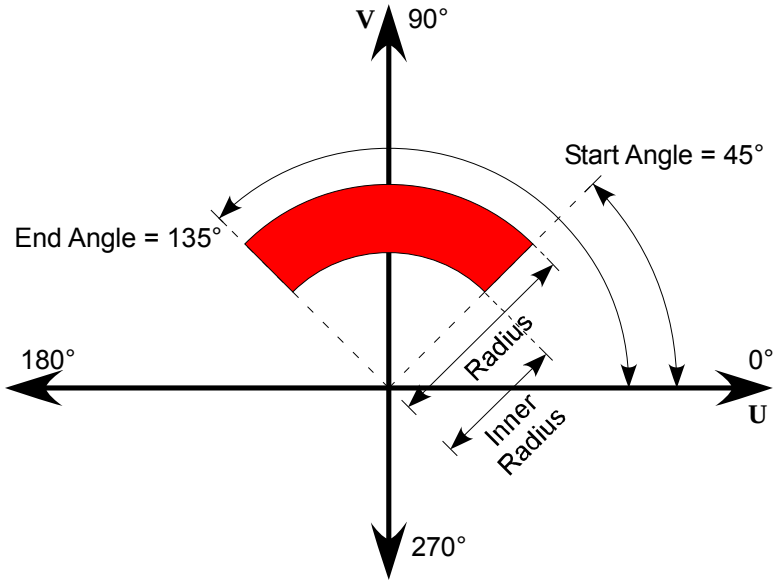


Figure 99 – Example of a Filled Circle (Arc) Symbol

Note that all circles and arcs are drawn counter-clockwise. If an arc's *Start Angle* is greater than its *End Angle*, then the arc will cross the +**U** axis as shown in Figure 100:

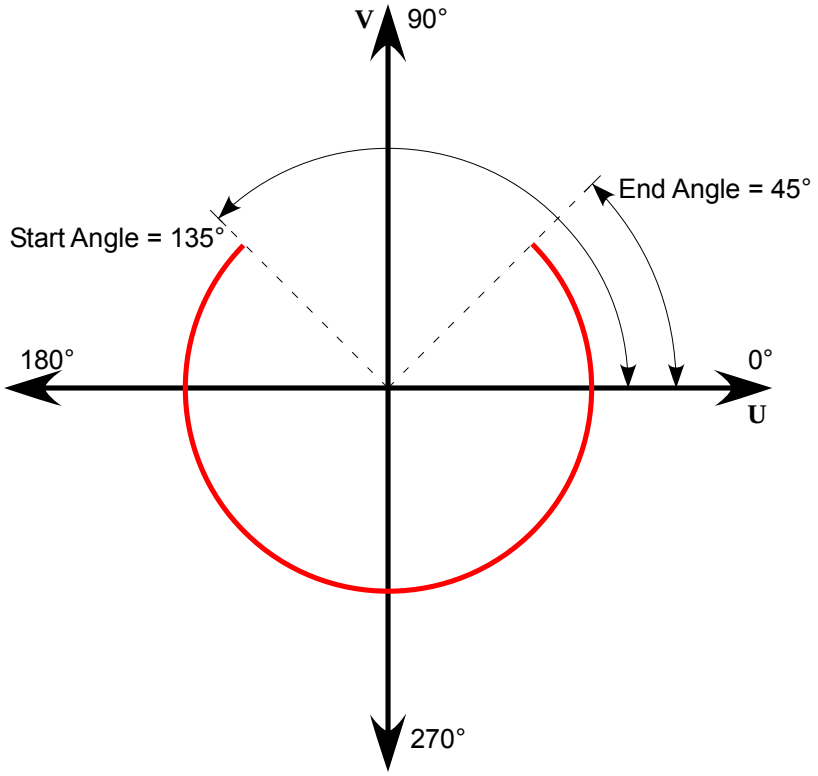


Figure 100 – Example of an Arc Crossing the +U Axis

When the IG creates a new symbol, that symbol is always hidden by default. The symbol is not made visible until the Host sends a **Symbol Control** packet or **Short Symbol Control** packet with 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 ID = 31								*Packet Size* = 16 + (24 × *n*)								*Symbol ID*															
Reserved							*1	Reserved								*Stipple Pattern*															
Line Width																															
Stipple Pattern Length																															
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 101 – Symbol Circle Definition Packet Structure

Table 38 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 38 – Symbol Circle Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 31	This parameter identifies this data packet as the **Symbol Circle Definition** packet. The value of this parameter must be 31.

Parameter	Description
Packet Size Type: unsigned int8 Units: bytes Value: $16 \le [16 + (24 \times n)] \le 232$ where n is the number of circles described by this packet	This parameter indicates the number of bytes in this data packet. This size must include the first 16 bytes of the packet plus all bytes following that describe individual circles. The minimum size of the **Symbol Circle Definition** packet is 16 bytes. The maximum packet size is 232 bytes, allowing for up to nine (9) individual circles. Although 10 circles could be defined using a total of 256 bytes, 255 is the largest value that can be represented by an unsigned int8. Note: Because all packets must begin and end on a 64-bit boundary, the value of this parameter must be an even multiple of eight (8).
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier must 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 circle or arc, then that symbol and any children will be destroyed and a new symbol created.
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.
Stipple Pattern Type: unsigned int16 Units: N/A	This parameter specifies the dash pattern used when drawing the curved line of a circle or arc. Each curved line is divided into sections that are $^{1}/_{32}$ of the length specified by the *Stipple Pattern Length* parameter. The stipple pattern is a bit mask that is used when drawing the sections. If a bit is set (1) then section corresponding to that bit will be drawn; if the bit is cleared (0) then the corresponding section will not be drawn. If the value of this parameter is 0xFFFF, then the line will be solid. If the line is longer than the stipple pattern length, the pattern is repeated. This value is ignored if the *Drawing Style* parameter is set to Fill (1).

Parameter	Description
Line Width Type: single float Units: Scaled symbol surface units	This parameter specifies the thickness of the line used to draw the circles and arcs. This thickness is measured in symbol surface units and will be scaled if the symbol is scaled (see Section 3.4.5.2). Note that if the symbol surface's horizontal and vertical units are not the same size, then horizontal, diagonal, and vertical lines will not appear to be the same thickness. This parameter is ignored if the *Drawing Style* parameter 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 is measured in symbol surface units and will be scaled if the symbol is scaled (see Section 3.4.5.2). If a line is longer than the stipple pattern length, then the pattern is repeated along that line. This parameter is ignored 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 with respect to the symbol's local coordinate system. This position is measured in scaled symbol surface units (see Section 3.4.5.2).
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 with respect to the symbol's local coordinate system. This position is measured in scaled symbol surface units (see Section 3.4.5.2).
Radius Type: single float Units: Scaled symbol surface units Values: ≥ 0.0 Datum: Center of circle	For a filled circle or arc, this parameter specifies the distance from the center of the circle to its outer circumference. For a line circle or arc, this parameter specifies the distance from the center of the circle to the center of the curve. This value is measured in scaled symbol surface units (see Section 3.4.5.2).

Parameter	Description
Inner Radius Type: single float Units: scaled symbol surface units Value: ≥ 0.0 to < *Radius* Datum: Center of circle	For a filled circle or arc, this parameter specifies the distance from the center of the circle to its inner boundary in scaled symbol surface units (see Section 3.4.5.2). The fill extends from the Inner Radius to the Radius. For line circles and arcs, this parameter is ignored.
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 and is measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc will 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 and is measured counter-clockwise from the +**U** axis. If *Start Angle* is greater than *End Angle*, then the arc will cross the +**U** axis.

4.1.32 Symbol Line Definition

The **Symbol Line Definition** packet is used to define a set of line segments or points. This packet can 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."

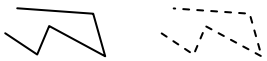
Each line symbol is identified by a *Symbol ID* value that is unique from all other symbols (including text and circle symbols). Every symbol must be created independently with its own unique *Symbol ID*, even if two or more symbols are visually identical.

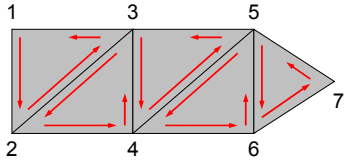
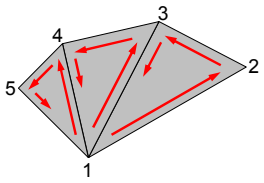
Once a **Symbol Line Definition** packet describing a circle symbol is sent to the IG, that symbol's type may not be changed. If a **Symbol Text Definition**, **Symbol Circle Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing circle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

Every line symbol is defined as an ordered set of zero or more points. Each point is defined with respect to the symbol's 2D coordinate system (see Section 3.4.5.2) 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 39:

Table 39 – 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 is 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.	
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 are disregarded.	

Primitive Type	Description	Example
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 vertices should be ordered such that all triangles are drawn counter-clockwise from the perspective of the view's eyepoint so that the triangles are front-facing.	 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 vertices should be arranged such that all triangles are drawn counter-clockwise from the perspective of the view's eyepoint so that the triangles are front-facing.	 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 will 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 will be drawn; if the bit is cleared (0) then the corresponding section will not be drawn. If the value of this parameter is 0xFFFF, then the line is solid.

The length of each section is equal to $^1/_{32}$ 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 is repeated.

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

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

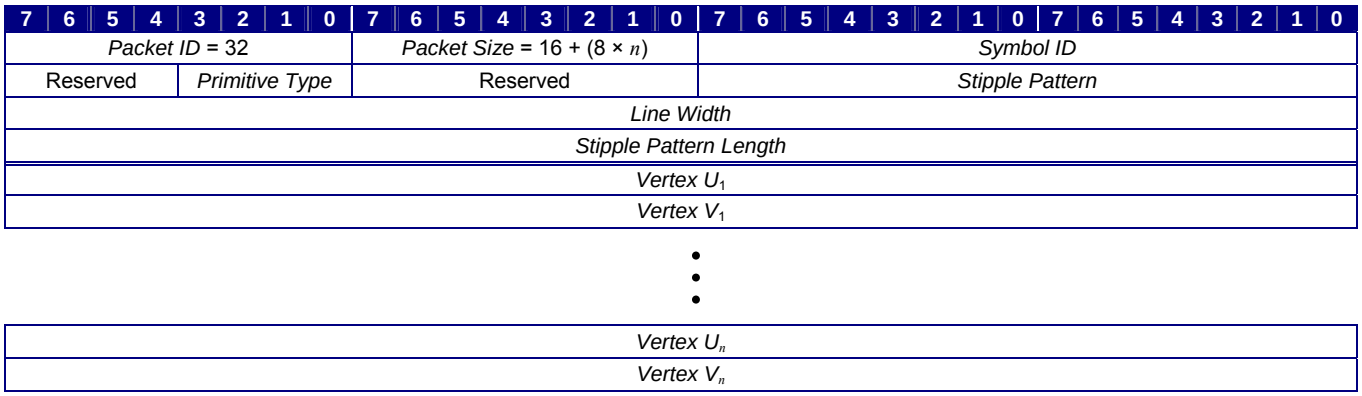


Figure 102 – Symbol Line Definition Packet Structure

Table 38 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 Line Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 32	This parameter identifies this data packet as the **Symbol Line Definition** packet. The value of this parameter must be 32.
Packet Size Type: unsigned int8 Units: bytes Value: $16 \le [16 + (8 \times n)] \le 248$ where n is the number of vertices described by this packet	This parameter indicates the number of bytes in this data packet. This size must include the first 16 bytes of the packet plus all bytes following that describe individual vertices. The minimum size of the **Symbol Line Definition** packet is 16 bytes. The maximum packet size is 248 bytes, allowing for up to 29 vertices. Although 30 vertices could be defined using a total of 256 bytes, 255 is the largest value that can be represented by an unsigned int8. Note: Because all packets must begin and end on a 64-bit boundary, the value of this parameter must be an even multiple of eight (8).

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier must 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 that symbol and any children will be destroyed and a new symbol created.
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 used in this symbol. The possible primitives are described in Table 39.
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 $^{1}/_{32}$ of the length specified by the *Stipple Pattern Length* parameter. The stipple pattern is a bit mask that is used when drawing the sections. If a bit is set (1) then section corresponding to that bit will be drawn; if the bit is cleared (0) then the corresponding section will not be drawn. If the value of this parameter is 0xFFFF, then the line will be solid. If the line is longer than the stipple pattern length, the pattern is repeated. This value is ignored if the *Primitive Type* parameter is set to Point (0), Triangle (4), Triangle Strip (5), or Triangle Fan (6).

Parameter	Description
Line Width Type: single float Units: Scaled symbol surface units	For point primitives, this parameter specifies the diameter of each point in the symbol. For line, line strip, and line loop primitives, this parameter specifies the thickness of each line in the symbol. The value of this parameter is measured in symbol surface units and will be scaled if the symbol is scaled (see Section 3.4.5.2). Note that if the symbol surface's horizontal and vertical units are not the same size, then horizontal, diagonal, and vertical lines will not appear to be the same thickness. This value is ignored 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 is measured in symbol surface units and will be scaled if the symbol is scaled (see Section 3.4.5.2). If a line is longer than the stipple pattern length, then the pattern is repeated along that line. This parameter is ignored if the *Drawing Style* parameter is set to Fill (1).
Vertex U Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the u position of a vertex with respect to the symbol's local coordinate system. This position is measured in scaled symbol surface units (see Section 3.4.5.2).
Vertex V Type: single float Units: Scaled symbol surface units Datum: Symbol 2D coordinate system	This parameter specifies the v position a vertex with respect to the symbol's local coordinate system.. This position is measured in scaled symbol surface units (see Section 3.4.5.2).

4.1.33 Symbol Clone

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

Alternatively, the **Symbol Clone** packet can be used to instantiate an IG-defined symbol template (see Section 3.3.3). Operations performed on the symbol instance will 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 Line Definition**, or **Symbol Clone** packet is received specifying the same *Symbol ID* but a different type, then the existing symbol will be destroyed along with any children and a new symbol will 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 will remain hidden until its state is changed with a **Symbol Control** packet.

The contents of the **Symbol Clone** 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 ID = 33								*Packet Size* = 8								*Symbol ID*															
Reserved							*1	Reserved								*Source ID*															

*1 *Source Type*

Figure 103 – Symbol Clone Definition 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 Clone Packet Structure

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 33	This parameter identifies this data packet as the **Symbol Clone** packet. The value of this parameter must be 33.
Packet Size Type: unsigned int8 Units: bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the identifier of the symbol that is being defined. This identifier must 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 that symbol and any children will be destroyed and a new symbol created.
Source Type Type: 1-bit field Units: N/A Values: 0 Symbol 1 Symbol Template	This parameter determines whether the new symbol will be a copy of an existing symbol or an instance of an IG-defined symbol template.
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 this parameter will specify the identifier of the symbol to be copied. If *Source Type* is set to Symbol Template (1), then this parameter will specify the identifier of the symbol template to be instantiated. If the specified source does not exist, then the packet will be ignored.

4.1.34 Symbol Control

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

A symbol must be defined before the Host sends a **Symbol Control** packet referencing that symbol. Symbols may be predefined by the IG or may be created by the Host sending any one of the symbol definition packets.

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

Symbols can be attached to one another in a hierarchical relationship. In such a hierarchy, a child symbol's position and rotation are 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. If this parameter is set to Attach (1), the symbol is attached to the symbol specified by the *Parent Symbol ID* parameter.

Figure 104 illustrates how symbols can 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.

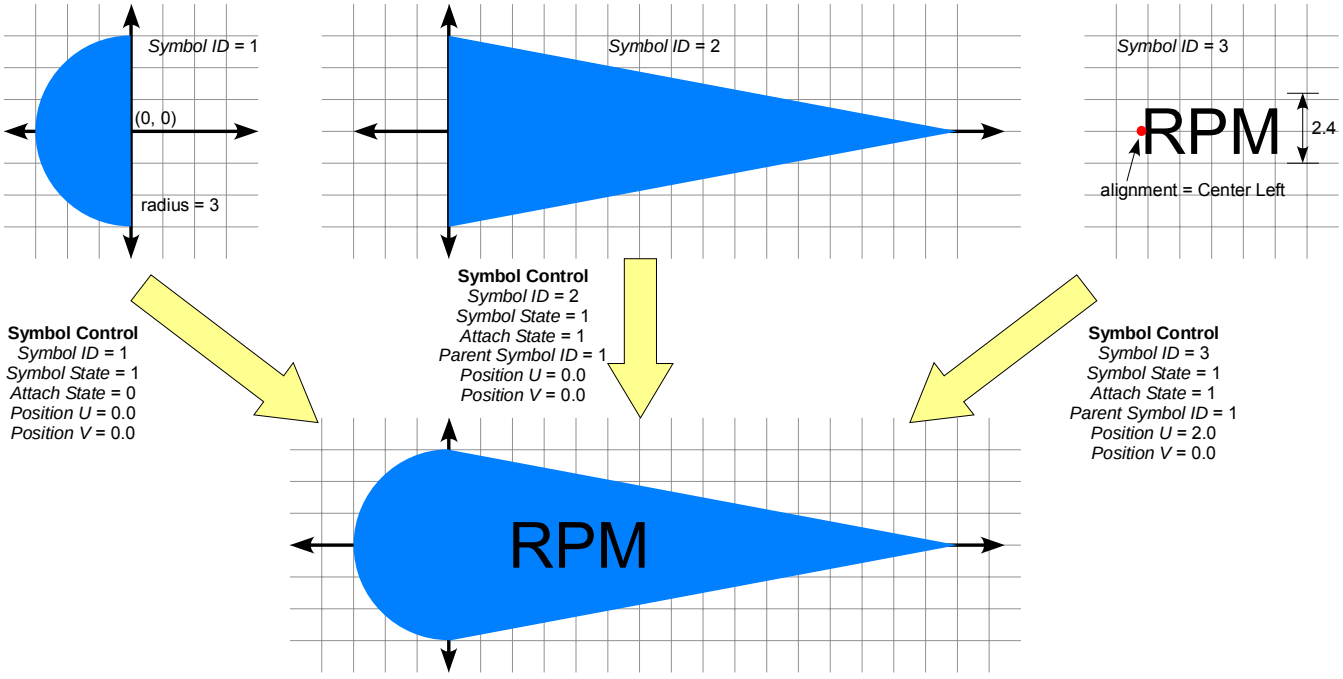


Figure 104 – Example of a Symbol Hierarchy

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.

The entire hierarchy can be rotated by rotating the circle symbol. As the circle symbol's local coordinate system rotates, all of its descendents are also rotated as shown in Figure 105:

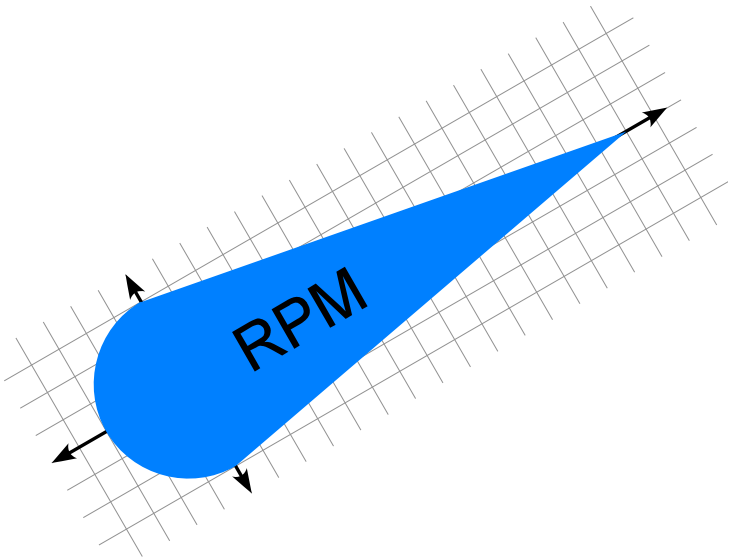


Figure 105 – Rotation of a Parent Symbol and its Children

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

The *Red*, *Blue*, *Green*, and *Alpha* parameters define 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 are ignored and the values of the parent are used. The *Inherit Color* parameter is 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 between two consecutive flashes. The *Flash Duty Cycle Percentage* parameter specifies the percentage of each flash cycle that the symbol is visible. If this parameter is set to 100%, then no flashing occurs and the symbol is always visible.

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

If a symbol's flash period or duty cycle is changed, then that symbol's flash cycle will 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 will not be reset. If the *Flash Control* parameter is set to Reset (1), then the symbol's flash cycle will be restarted from the beginning.

Figure 106 illustrates a typical flash period and duty cycle. The period is 0.4 seconds and the duty cycle is 75%. The cycle repeats itself until it is restarted or the symbol is destroyed.

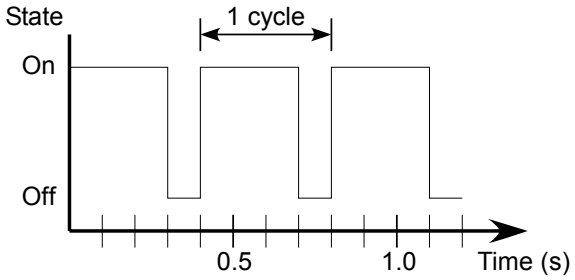


Figure 106 – Example of a Symbol Flash Cycle

The drawing order of symbols is determined by layer. Each symbol surface has 256 logical layers, which are rendered in order of increasing layer number. In other words, symbols assigned to Layer 0 are 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 will be drawn in order of increasing symbol ID.

The position of a top-level (root) symbol is always specified with respect to the surface's 2D coordinate system (see Section 3.4.5.1). The position of a child symbol is always specified with respect to the parent symbol's local coordinate system (see Section 3.4.5.2).

The *Scale U* and *Scale V* parameters specify a symbol's scale along the symbol's local **U** and **V** axes, respectively. The symbol's apparent size is also affected by any ancestors' scaling factors as described in Section 3.4.5.2.

Once a **Symbol Control** packet is sent to the IG, the state of the specified symbol will 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 ID = 34								*Packet Size* = 40								*Symbol ID*															
*5			*4	*3	*2	*1		Reserved								*Parent Symbol ID*															
Surface ID																*Layer*								*Flash Duty Cycle Percentage*							
Flash Period																															
Position U																															
Position V																															
Rotation																															
Red								*Green*								*Blue*								*Alpha*							
Scale U																															
Scale V																															

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

Figure 107 – 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 – Symbol Line Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 34	This parameter identifies this data packet as the **Symbol Control** packet. The value of this parameter must be 34.
Packet Size Type: unsigned int8 Units: bytes Value: 40	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 40.
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol to which this packet is applied. This value must be unique for each active symbol.
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. This parameter may be set to one of the following values: **Hidden** – The symbol is hidden from view; however, it can be positioned, rotated, and scaled. It can also be attached to another symbol as a child. It can also be used as a parent by other symbols, although any children are also hidden. **Visible** – The symbol is drawn on the surface. It can be positioned, rotated, and scaled. It can also be attached to another symbol as a child. It can also be used as a parent by other symbols. **Destroyed** – The symbol is deleted and any system resources are freed. Any children are also destroyed. All other parameters in this packet are ignored. 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 are ignored.

Parameter	Description
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the symbol should be attached as a child to a parent symbol. If this parameter is set to Detach (0), then the symbol becomes or remains a top-level (non-child) symbol. The *Parent Symbol* parameter is ignored. The *U Position*, *V Position*, and *Rotation* parameters specify the symbol's position and rotation relative to the symbol surface's local coordinate system (see Section 3.4.5.1). If this parameter is set to Attach (1), then the symbol becomes or remains attached to the symbol specified by the *Parent Symbol ID* parameter. The *U Position*, *V Position*, and *Rotation* parameters specify the symbol's position and rotation relative to the parent symbol's local coordinate system (see Section 3.4.5.2). The attach state of a symbol may be changed at any time. The attachment or detachment takes place immediately and remains 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 flash cycle is continued from its present point or whether it is restarted at the beginning. This parameter is ignored if either *Flash Duty Cycle Percentage* or *Flash Period* is changed. This parameter may also be ignored 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, is 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), this parameter is ignored.
Parent Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the parent for the symbol. If the *Attach State* parameter is set to Detach (0), this parameter is ignored. The value of this parameter may be changed without first detaching the symbol from its existing parent. If the specified parent symbol is invalid, no change in the attachment will be made.

Parameter	Description
Surface ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol surface on which the symbol is drawn. If the symbol is a child, then the top-level parent symbol's surface is used and this parameter is ignored. If the specified surface is invalid and this symbol is not a child, then the symbol will not be drawn.
Layer Type: unsigned int8 Units: N/A Values: 0 – 255	This parameter specifies the layer to which the symbol is assigned. Layers are drawn in order of increasing layer number. For example, Layer 0 will 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 symbols are drawn in order of increasing symbol ID. Note that 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 is the percentage of one flash cycle that the symbol will be visible. If this value is set to zero (0), then the symbol is always invisible. If this value is set to 100%, then the symbol is always visible. This parameter is ignored if this symbol inherits flashing behavior.
Flash Period Type: single float Units: seconds	This parameter specifies the duration of a single flash cycle. This parameter is ignored if *Flash Duty Cycle Percentage* is set to 0% or 100%. This parameter is ignored if the symbol inherits its flashing behavior from its parent.
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, this position is defined with respect to the symbol surface's 2D coordinate system as described in Section 3.4.5.1. For child symbols, this position is defined with respect to the parent symbol's local coordinate system as described in Section 3.4.5.2.

Parameter	Description
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, this position is defined with respect to the symbol surface's 2D coordinate system as described in Section 3.4.5.1. For child symbols, this position is defined with respect to the parent symbol's local coordinate system as described in Section 3.4.5.2.
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 a rotation for the symbol. This rotation is always counter-clockwise about the symbol's local origin. Note that each child symbol is oriented relative to its parent's local coordinate system.
Red Type: unsigned int8 Units: N/A	This parameter specifies the red component of the symbol's color. This value is ignored if *Inherit Color* is set to inherit (1).
Green Type: unsigned int8 Units: N/A	This parameter specifies the green component of the symbol's color. This value is ignored if *Inherit Color* is set to inherit (1).
Blue Type: unsigned int8 Units: N/A	This parameter specifies the blue component of the symbol's color. This value is ignored if *Inherit Color* is set to inherit (1).
Alpha Type: unsigned int8 Units: N/A	This parameter specifies the alpha component of the symbol's color. A value of zero (0) corresponds to fully transparent; a value of 255 corresponds to fully opaque. This value is ignored if *Inherit Color* is set to inherit (1).

Parameter	Description
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. A value less than 1.0 will cause the symbol to be reduced in size. A value greater than 1.0 will cause the symbol to be increased in size. Note that a symbol's actual size and shape will be affected by the symbol's scaling factors, as well as those of any ancestor symbols.
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. A value less than 1.0 will cause the symbol to be reduced in size. A value greater than 1.0 will cause the symbol to be increased in size. Note that a symbol's actual size and shape will be affected by the symbol's scaling factors, as well as those of any ancestor symbols.

4.1.35 Short Symbol Control

The **Short Symbol Control** packet is provided as a lower-bandwidth alternative to the **Symbol Control** packet (Section 4.1.33). It can be used when manipulation of only one or two symbol attributes of a symbol are 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 determine what data types are used to interpret the *Attribute Value 1* and *Attribute Value 2* parameters, respectively.

A symbol must be defined before the Host sends a **Short Symbol Control** packet referencing that symbol. Symbols may be predefined by the IG or may be created by the Host sending any one of the symbol definition packets.

Before the Host can send a **Short Symbol Control** referencing a symbol, the Host must first send a **Symbol Control** packet referencing that symbol so that all of the symbol's attributes can be set.

The contents of the **Short 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 ID = 35								*Packet Size* = 32								*Symbol ID*															
*5			*4	*3	*2	*1		Reserved								*Attribute Select 1*								*Attribute Select 2*							
Attribute Value 1																															
Attribute Value 2																															

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

Figure 108 – Short Symbol Control 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 43 – Symbol Line Definition Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 35	This parameter identifies this data packet as the **Short Symbol Control** packet. The value of this parameter must be 35.
Packet Size Type: unsigned int8 Units: bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
Symbol ID Type: unsigned int16 Units: N/A	This parameter specifies the symbol to which this packet is applied. This value must be unique for each active symbol.
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. This parameter may be set to one of the following values: **Hidden** – The symbol is hidden from view; however, it can be positioned, rotated, and scaled. It can also be attached to another symbol as a child. It can also be used as a parent by other symbols, although any children are also hidden. **Visible** – The symbol is drawn on the surface. It can be positioned, rotated, and scaled. It can also be attached to another symbol as a child. It can also be used as a parent by other symbols. **Destroyed** – The symbol is deleted and any system resources are freed. Any children are also destroyed. All other parameters in this packet are ignored.
Attach State Type: 1-bit field Units: N/A Values: 0 Detach 1 Attach	This parameter specifies whether the symbol should be attached as a child to a parent symbol. If this parameter is set to Detach (0), then the symbol becomes or remains a top-level (non-child) symbol. If this parameter is set to Attach (1), then the symbol becomes or remains attached to the specified parent. This parent must be specified by either the *Attribute Value 1* or the *Attribute Value 2* parameter, and the corresponding *Attribute Select* parameter must be set to Parent Symbol ID (2).
Flash Control Type: 1-bit field Units: N/A Values: 0 Continue 1 Reset	This parameter specifies whether the flash cycle is continued from its present point or whether it is restarted at the beginning. This parameter is ignored if this packet is being used to change either the symbol's *Flash Duty Cycle Percentage* or *Flash Period* attributes. This parameter may also be ignored if the *Flash Duty Cycle Percentage* attribute is set to 0 or 100.

Parameter	Description
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, is 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), this parameter is ignored.
Attribute Select 1 Type: 2-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* is ignored.
Attribute Select 2 Type: 2-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* is ignored.

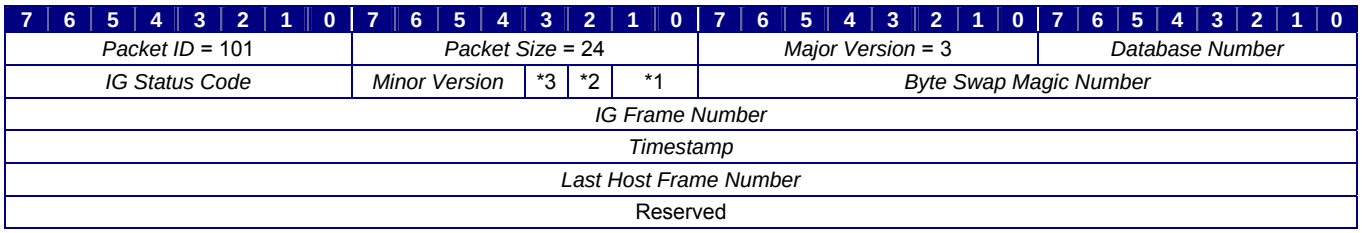
Parameter	Description
Attribute Value 1 Type: Attribute-specific Units: Attribute-specific	This parameter specifies the value of the attribute identified by the *Attribute Select 1* field. If *Attribute Select 1* is set to Surface ID (1), Parent Symbol ID (2), Layer (3), or Flash Duty Cycle Percentage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

4.2 IG-to-Host Packets

4.2.1 Start of Frame

The **Start of Frame** packet is used to signal the beginning of a new frame. Every IG-to-Host message must contain *exactly one* **Start of Frame** packet. This packet must be the *first* packet in the message.

The contents of the **Start of Frame** packet are as follows:



*1 *IG Mode*
*2 *Timestamp Valid*
*3 *Earth Reference Model*

Figure 109 – Start of Frame 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 – Start of Frame Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 101	This parameter identifies this data packet as the **Start of Frame** packet. The value of this parameter must be 101.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
Major Version Type: unsigned int8 Units: N/A Value: 3	This parameter indicates the major version of the CIGI interface that is currently being used by the IG. The Host can use this number to determine concurrency. The value of this parameter must be 3.

Parameter	Description
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 into primary memory. The Host will set the *Database Number* parameter of the **IG Control** packet to direct the IG to begin loading the corresponding database. The IG will 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 will 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 will set this parameter to the positive database number. The IG can now be considered mission-ready. If the Host requests a database that does not exist or cannot be loaded, the IG will set this parameter to -128. Zero (0) is used to indicate that the IG controls the database loading. Refer to Section 4.1.1 for more information about the **IG Control** packet.
IG Status Code Type: unsigned int8 Units: N/A Values: 0 Normal operation 1 – 255 Defined by IG	This parameter indicates the error status of the IG. Error codes are IG-specific. Refer to the appropriate IG documentation for a list of error codes. If more than one error is detected, the IG will report the one with the highest priority. If additional error reporting must be performed, the IG should be placed in Debug mode via the **IG Control** packet's *IG Mode* parameter or the IG's user interface.

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 may be one of the following values: **Reset/Standby** – This is the IG's initial state upon start-up. When set to this mode, the IG will initialize/reinitialize the simulation. All entities that were instantiated during a previous mission will be destroyed. All environmental properties, views, components, and sensors will revert to their default settings. Any Host-defined rates, trajectories, and collision detection segments and volumes will be removed. The IG will only send the Start of Frame data packet to the Host and will ignore Host inputs except for the *IG Mode* parameter of the **IG Control** data packet. The IG will 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 will 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 ignores all data from the Host and sends only **Start of Frame** packets. This mode can be activated only from the IG. Because the **IG Control** packets from the Host are ignored by the IG, the IG must be placed into Reset/Standby mode before the Host can 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. Because the *Timestamp* parameter is required for asynchronous operation, *Timestamp Valid* must be set to Valid (1) in this 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 4.1.19). If the Host defines an ERM that the IG cannot support, this value is set to WGS 84 (0). In such cases, the Host must redefine the ERM or use the WGS 84 reference ellipsoid.
Minor Version Type: 4-bit field Units: N/A Value: 2	This parameter indicates the minor version of the CIGI interface that is currently being used by the IG. The Host can use this number to determine concurrency.
Byte Swap Magic Number Type: unsigned int16 Units: N/A Values: 8000h No byte swap (see note at right) 80h Byte swap	This parameter is used by the Host to determine whether it needs to byte-swap incoming data. Refer to Section 2.5 for details on this mechanism. Note: The IG *must* set this value to 8000h, or 32768.
IG Frame Number Type: unsigned int32 Units: N/A	This parameter uniquely identifies an IG data frame. The IG should 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.

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. This will enable the IG to correct for latencies as described in Section 2.2.1. 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 use of this parameter is required for asynchronous operation. The use of this parameter is optional for synchronous operation. If this parameter does not contain a valid timestamp, the *Timestamp Valid* parameter should be set to zero (0).
Last Host Frame Number Type: unsigned int32 Units: N/A	This parameter contains 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.

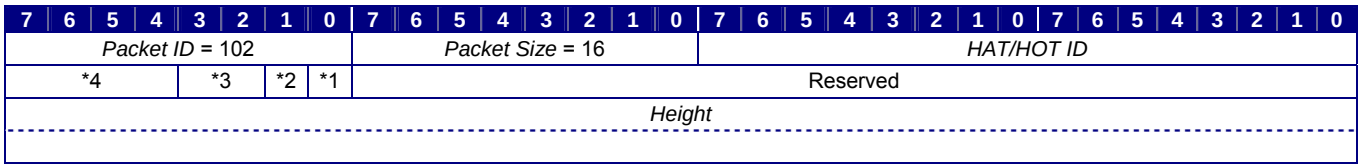
4.2.2 HAT/HOT Response

The **HAT/HOT Response** packet is sent by the IG in response to a **HAT/HOT Request** packet (Section 4.1.24) 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 *Host Frame Number LSN* parameter of each corresponding **HAT/HOT Response** packet must contain the least significant nybble 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 will be set to Invalid (0).

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



*1 *Valid*
*2 *Response Type*
*3 Reserved
*4 *Host Frame Number LSN*

Figure 110 – HAT/HOT Response Packet Structure

Table 45 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 45 – HAT/HOT Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 102	This parameter identifies this data packet as the **HAT/HOT Response** packet. The value of this parameter must be 102.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
HAT/HOT ID Type: unsigned int16 Units: N/A	This parameter identifies the HAT or HOT response. This value corresponds to the value of the *HAT/HOT ID* parameter in the associated **HAT/HOT Request** packet.
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Height* parameter contains a valid number. A value of zero (0) indicates that the test point was 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.
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 or HOT is calculated. This parameter is ignored if the *Update Period* parameter of the corresponding **HAT/HOT Request** packet was set to zero (0).
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 represents the Height Above Terrain. If *Request Type* is set to HOT (1), this value represents the Height Of Terrain. This parameter is valid only if the *Valid* parameter is set to one (1).

4.2.3 HAT/HOT Extended Response

The **HAT/HOT Extended Response** packet is sent by the IG in response to a **HAT/HOT Request** packet (Section 4.1.24) 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 3.4.1.2) as shown in the figure below. A vector length is not specified; the vector can be assumed to be a unit vector.

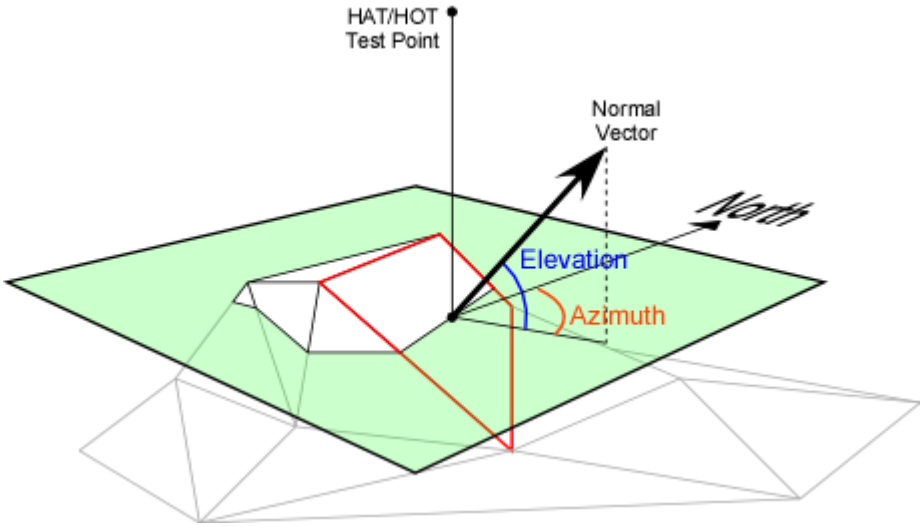


Figure 111 – HAT/HOT Extended Response Normal Vector

If the *Update Period* parameter of the originating **HAT/HOT Request** packet was set to a value greater than zero, then the *Host Frame Number LSN* parameter of each corresponding **HAT/HOT Response** packet must contain the least significant nybble 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 will be set to zero (0).

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 ID = 103								*Packet Size* = 40								*HAT/HOT ID*															
*2				Reserved			*1	Reserved																							
HAT																															
HOT																															
Material Code																															
Normal Vector Azimuth																															
Normal Vector Elevation																															
Reserved																															

*1 *Valid*
*2 *Host Frame Number LSN*

Figure 112 – HAT/HOT Extended Response Packet Structure

Table 46 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 46 – HAT/HOT Extended Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 103	This parameter identifies this data packet as the **HAT/HOT Extended Response** packet. The value of this parameter must be 103.
Packet Size Type: unsigned int8 Units: Bytes Value: 40	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 40.
HAT/HOT ID Type: unsigned int16 Units: N/A	This parameter identifies the HAT/HOT response. This value corresponds to the value of the *HAT/HOT ID* parameter in the associated **HAT/HOT Request** packet.
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. A value of zero (0) indicates that the test point was 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 or HOT is calculated. This parameter is ignored if the *Update Period* parameter of the corresponding **HAT/HOT Request** packet was set to zero (0).
HAT Type: double float Units: meters Datum: Terrain/Sea Surface Height	This parameter indicates the height of the test point above the terrain. A negative value indicates that the test point is below the terrain. This parameter is valid only if the *Valid* parameter is set to one (1).
HOT Type: double float Units: meters Datum: Mean Sea Level	This parameter indicates the height of terrain above or below the test point. This value is relative to the ellipsoid height, or Mean Sea Level. This parameter is valid only if the *Valid* parameter is set to one (1).
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. This parameter is valid only if the *Valid* parameter is set to one (1).
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. This value is the horizontal angle from True North to the normal vector. This parameter is valid only 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 (Section 3.4.1.2)	This parameter indicates the elevation of a vector normal to the surface intersected by the HAT/HOT test vector. This value is the vertical angle from the geodetic reference plane to the normal vector. This parameter is valid only if the *Valid* parameter is set to one (1).

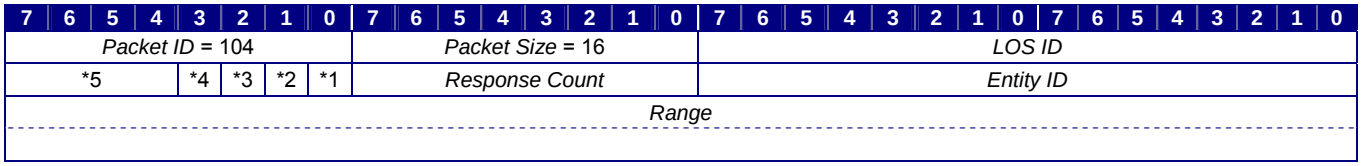
4.2.4 Line of Sight Response

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 the Line of Sight (LOS) segment or vector source point to the point of intersection with a polygon surface. The packet is sent when the *Request Type* parameter of the request packet is set to Basic (0).

A **Line of Sight Response** packet will be sent for each intersection along the LOS segment or vector. The *Response Count* parameter will contain the total number of responses that are being returned. This will allow the Host to determine when all response packets for the given request have been received.

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 *Host Frame Number LSN* parameter of each corresponding **Line of Sight Response** packet must contain 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 Response** packet are as follows:



*1 *Valid*
*2 *Entity ID Valid*
*3 *Visible*
*4 Reserved
*5 *Host Frame Number LSN*

Figure 113 – Line of Sight Response 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 – Line of Sight Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 104	This parameter identifies this data packet as the **Line of Sight Response** packet. The value of this parameter must be 104.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS response. This value corresponds to the value of the *LOS ID* parameter in the associated **Line of Sight Segment Request** packet (Section 4.1.25) or **Line of Sight Vector Request** packet (Section 4.1.26).
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether the *Range* parameter is valid. The range will be invalid if no intersection occurs, or if an intersection occurs before the minimum range or beyond the maximum range specified in a LOS vector request.
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 (Valid) or a non-entity (Invalid).
Visible Type: 1-bit field Units: N/A Values: 0 Occluded (not visible) 1 Visible	This parameter is used in response to a **Line of Sight Segment Request** packet. It indicates whether the destination point is visible from the source point. This value should be ignored if the packet is in response to a **Line of Sight Vector Request** packet Note: If the LOS segment destination point is within the body of a target entity model, this parameter will be set to Occluded (0) and the *Entity ID* parameter will contain the ID of that entity.
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. This parameter is ignored 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 will return for the corresponding request. Note: If *Visible* is set to Visible (1), then *Response Count* should be set to 1.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity with which an LOS test vector or segment intersects. This parameter should be ignored if *Entity ID Valid* is set to Invalid (0).
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.

4.2.5 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 4.2.4 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 219, Figure 111). The packet is sent when the *Request Type* parameter of the request packet is set to Extended (1).

A **Line of Sight Extended Response** packet will be sent for each intersection along the LOS segment or vector. The *Response Count* parameter will contain the total number of responses that are being returned. This will allow 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 114 illustrates two LOS test segments and the data returned with the responses:

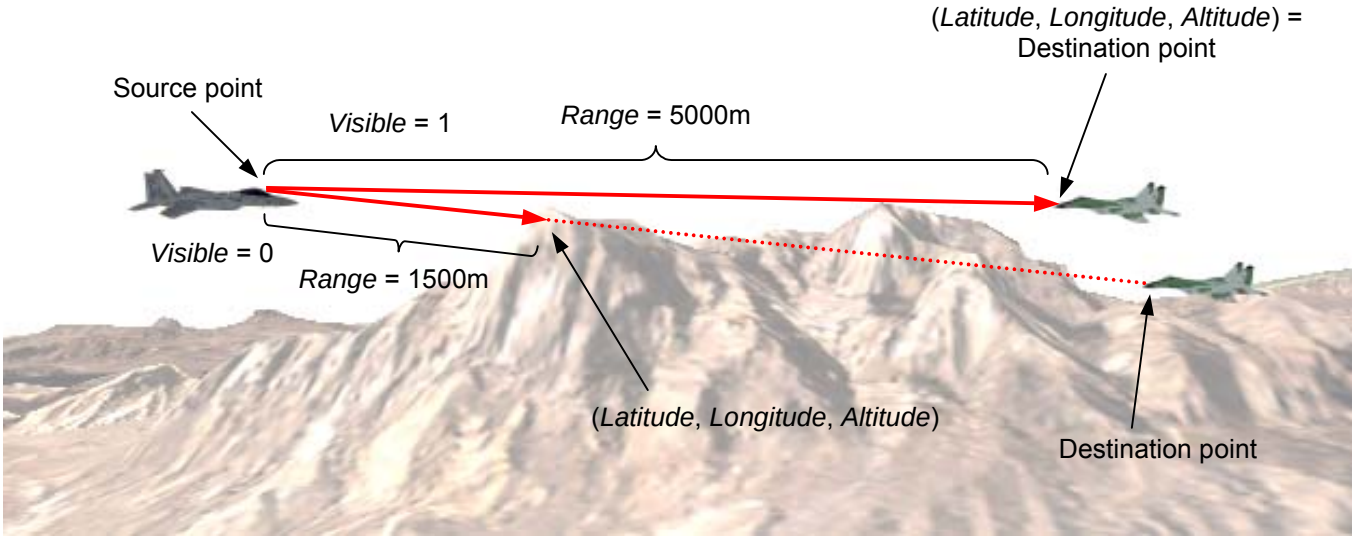


Figure 114 – Responses to Line of Sight Segment Requests

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 115 illustrates two LOS test vectors and the data returned with the responses:

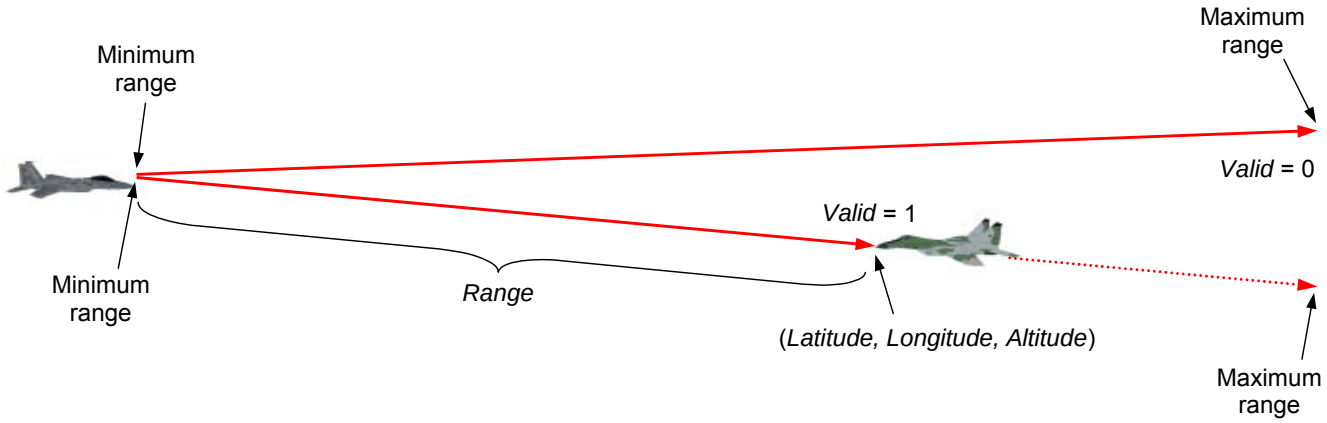


Figure 115 – Responses to Line of Sight Vector Requests

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 *Host Frame Number LSN* parameter of each corresponding **Line of Sight Extended Response** packet must contain the least significant nybble of the *Host Frame Number* value last received by the IG before the line of sight value 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 ID = 105								*Packet Size* = 56								*LOS ID*															
*5				*4	*3	*2	*1	*Response Count*								*Entity ID*															
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 116 – Line of Sight Extended Response 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 – Line of Sight Extended Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 105	This parameter identifies this data packet as the **Line of Sight Extended Response** packet. The value of this parameter must be 105.
Packet Size Type: unsigned int8 Units: Bytes Value: 56	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 56.
LOS ID Type: unsigned int16 Units: N/A	This parameter identifies the LOS response. This value corresponds to the value of the *LOS ID* parameter in the associated **Line of Sight Segment Request** or **Line of Sight Vector Request** packet.
Valid Type: 1-bit field Units: N/A Values: 0 Invalid 1 Valid	This parameter indicates whether this packet contains valid data. A value of Invalid (0) indicates that the LOS test segment or vector did not intersect any geometry.
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 (Valid) or a non-entity (Invalid).
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 will be invalid if an intersection occurs before the minimum range or beyond the maximum range specified in an LOS vector request. The range will also be invalid if this packet is in response to an LOS segment request. If *Valid* is set to Invalid (0), this parameter will also be set to Invalid (0).

Parameter	Description
Visible Type: 1-bit field Units: N/A Values: 0 Occluded (not visible) 1 Visible	This parameter is used in response to a **Line of Sight Segment Request** packet and indicates whether the destination point is visible from the source point. This value should be ignored if the packet is in response to a **Line of Sight Vector Request** packet Note: If the LOS segment destination point is within the body of a target entity model, this parameter will be set to Occluded (0) and the *Entity ID* parameter will contain the ID of that entity.
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. This parameter is ignored 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 Extended Response** packets the IG will return for the corresponding request. Note: If *Visible* is set to Visible (1), then *Response Count* should be set to 1.
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity with which a LOS test vector or segment intersects. This parameter should be ignored if *Entity ID Valid* is set to Invalid (0).
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 an object.

Parameter	Description
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), this parameter indicates 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), this parameter specifies the offset of the point of intersection of the LOS test segment or vector along the intersected entity's **X** axis.
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), this parameter indicates 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), this parameter specifies the offset of the point of intersection of the LOS test segment or vector along the intersected entity's **Y** axis.

Parameter	Description
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), this parameter indicates 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), this parameter specifies 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	This parameter indicates the red color component of the surface at the point of intersection.
Green Type: unsigned int8 Units: N/A	This parameter indicates the green color component of the surface at the point of intersection.
Blue Type: unsigned int8 Units: N/A	This parameter indicates the blue color component of the surface at the point of intersection.
Alpha Type: unsigned int8 Units: N/A	This parameter indicates the alpha component of the surface at the point of intersection.
Material Code Type: unsigned int32 Units: N/A	This parameter indicates the material code of the surface intersected by the LOS test segment or vector.

Parameter	Description
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. This value is the horizontal angle from True North to the normal vector. This parameter is valid only 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. This value is the vertical angle from the geodetic reference plane to the normal vector. This parameter is valid only if the *Valid* parameter is set to one (1).

4.2.6 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:

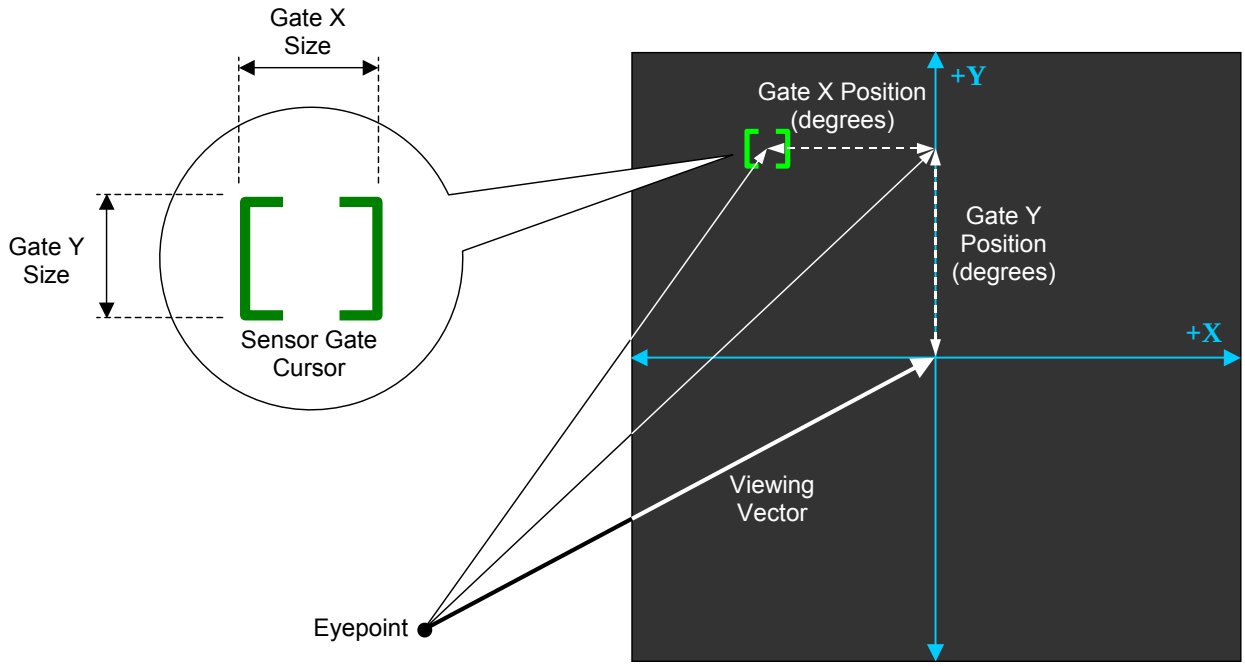


Figure 117 – Sensor Gate Size and Position

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 can be performed with a **View Definition** packet (Section 4.1.21).

The *Host Frame Number* parameter contains value of the *Host Frame Number* parameter of the **IG Control** packet (see Section 2.2.2) 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 4.2.7) must 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 ID = 106								*Packet Size* = 24								*View ID*															
Sensor ID								Reserved							*1	Reserved															
Gate X Size																*Gate Y Size*															
Gate X Position																															
Gate Y Position																															
Host Frame Number																															

*1 *Sensor Status*

Figure 118 – Sensor 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 – Sensor Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 106	This parameter identifies this data packet as the **Sensor Response** packet. The value of this parameter must be 106.
Packet Size Type: unsigned int8 Units: Bytes Value: 24	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 24.
View ID Type: unsigned int16 Units: N/A	This parameter specifies the view that represents the sensor display.
Sensor ID Type: unsigned int8 Units: N/A	This parameter specifies the sensor to which the data in this packet apply.

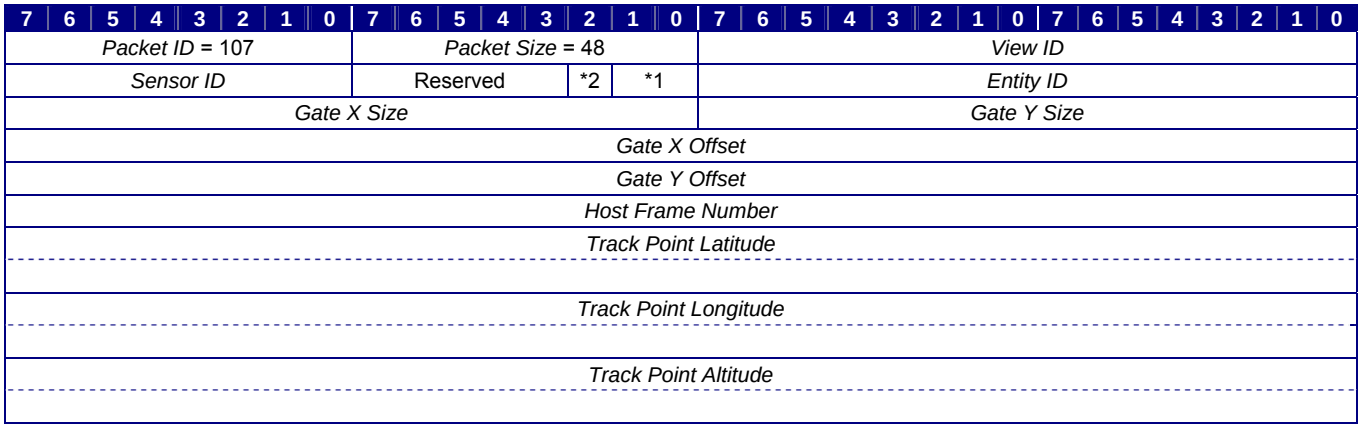
Parameter	Description
Sensor Status Type: unsigned 2-bit field Units: N/A Values: 0 Searching for target 1 Tracking target 2 Impending breaklock 3 Breaklock	This parameter indicates the current tracking state of the sensor.
Gate X Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	This parameter specifies 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)	This parameter specifies 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. This position is given as 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. This position is given as 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	This parameter indicates the Host frame number at the time that the IG calculates the gate and line-of-sight intersection data.

4.2.7 Sensor Extended Response

The **Sensor Extended Response** packet, like the **Sensor Response** packet (Section 4.2.6), 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 must 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:



*1 *Sensor Status*
*2 *Entity ID Valid*

Figure 119 – Sensor 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 – Sensor Extended Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 107	This parameter identifies this data packet as the **Sensor Extended Response** packet. The value of this parameter must be 107.
Packet Size Type: unsigned int8 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 48.

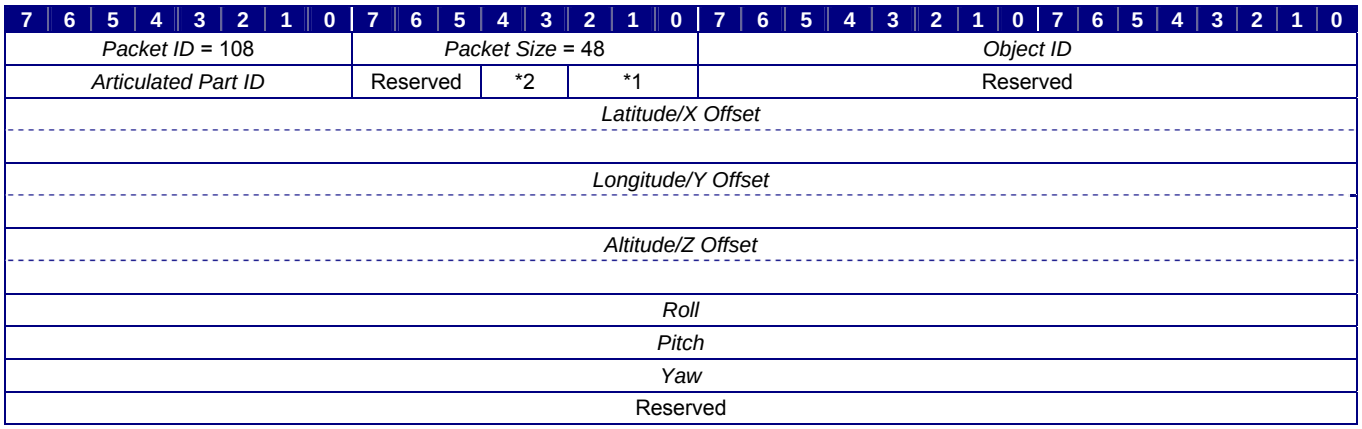
Parameter	Description
View ID Type: unsigned int16 Units: N/A	This parameter specifies the view that represents the sensor display.
Sensor ID Type: unsigned int8 Units: N/A	This parameter specifies the sensor to which the data in this packet apply.
Sensor Status Type: unsigned 2-bit field Units: N/A Values: 0 Searching for target 1 Tracking target 2 Impending breaklock 3 Breaklock	This parameter indicates 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. If this parameter is set to Valid (1), then *Entity ID* identifies the target entity.
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity ID of the target. This parameter is ignored if *Entity ID Valid* is set to Invalid (0).
Gate X Size Type: unsigned int16 Units: pixels or raster lines (see note at right)	This parameter specifies 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)	This parameter specifies 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. This position is given as 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. This position is given as 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	This parameter indicates the Host frame number at the time that 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	This parameter indicates the geodetic latitude of the point being tracked by the sensor. This parameter is valid only when the Sensor Status parameter is set to one (1) or two (2).
Track Point Longitude Type: double float Units: degrees Values: -180.0 – 180.0 Datum: Prime Meridian	This parameter indicates the geodetic longitude of the point being tracked by the sensor. This parameter is valid only when the Sensor Status parameter is set to one (1) or two (2).
Track Point Altitude Type: double float Units: meters Datum: Mean Sea Level	This parameter indicates the geodetic altitude of the point being tracked by the sensor. This parameter is valid only when the Sensor Status parameter is set to one (1) or two (2).

4.2.8 Position Response

The **Position Response** packet is sent by the IG in response to a **Position Request** packet (Section 4.1.27). 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:



*1 *Object Class*
*2 *Coordinate System*

Figure 120 – Position 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 – Position Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 108	This parameter identifies this data packet as the **Position Response** packet. The value of this parameter must be 108.
Packet Size Type: unsigned int8 Units: Bytes Value: 48	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 48.

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 reported. If *Object Class* is set to Articulated Part (1), this parameter indicates the entity whose part is identified by the *Articulated Part ID* parameter.
Articulated Part ID Type: unsigned int8 Units: N/A	This parameter identifies the articulated part whose position is being reported. The entity to which the part belongs is specified by the *Object ID* parameter. This parameter is valid only when *Object Class* is 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	This parameter indicates 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	This parameter indicates the coordinate system in which the position and orientation are specified. **Geodetic** – Position is specified as a geodetic latitude, longitude, and altitude. Orientation is given with respect to the reference plane shown in Figure 18, page 25. **Parent Entity** – Position and orientation are with respect to the entity to which the specified child entity, articulated part, view, or view group is attached. This value is invalid for top-level entities. **Submodel** – Position and orientation are with respect to the articulated part's reference coordinate system as described in Section 3.4.3. This value is valid only when *Object Class* is set to Articulated Part (1). Note: If *Object Class* is set to Motion Tracker (4), this parameter is ignored and the positional and rotational data are relative to the tracking device boresight state.

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), this parameter indicates the geodetic latitude of the entity, articulated part, 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), this parameter indicates 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), this parameter indicates the X offset from the articulated part submodel's reference point (see Section 3.4.3) If *Object Class* is set to Motion Tracker (4), this parameter indicates 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), this parameter indicates 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), this parameter indicates 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), this parameter indicates the Y offset from the articulated part submodel's reference point (see Section 3.4.3) If *Object Class* is set to Motion Tracker (4), this parameter indicates 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), this parameter indicates 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), this parameter indicates 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), this parameter indicates the Z offset from the articulated part submodel's reference point (see Section 3.4.3) If *Object Class* is set to Motion Tracker (4), this parameter indicates 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	This parameter indicates the roll angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), this parameter indicates 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	This parameter indicates the pitch angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), this parameter indicates 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	This parameter indicates the yaw angle of the specified entity, articulated part, view, or view group. If *Object Class* is set to Motion Tracker (4), this parameter indicates the yaw angle reported by the tracking device.

4.2.9 Weather Conditions Response

The **Weather Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 4.1.28) whose *Request Type* parameter specifies Weather Conditions. The packet describes atmosphere properties at the requested geodetic position.

Figure 121 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:

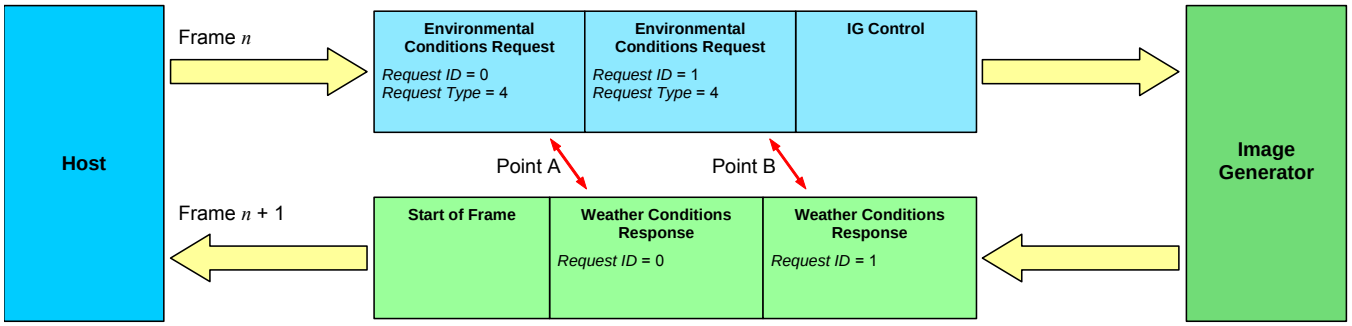


Figure 121 – Data Exchange for Weather Conditions Request

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 ID = 109	*Packet Size* = 32	*Request ID*	*Humidity*
Air Temperature			
Visibility Range			
Horizontal Wind Speed			
Vertical Wind Speed			
Wind Direction			
Barometric Pressure			
Reserved			

Figure 122 – Weather Conditions 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 – Weather Conditions Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 109	This parameter identifies this data packet as the **Weather Conditions Response** packet. The value of this parameter must be 109.
Packet Size Type: unsigned int8 Units: Bytes Value: 32	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 32.
Request ID Type: unsigned int8 Units: N/A	This parameter identifies the environmental conditions request to which this response packet corresponds.
Humidity Type: unsigned int8 Units: percent Values: 0 – 100	This parameter indicates the humidity at the requested location.
Air Temperature Type: single float Units: degrees Celsius (°C)	This parameter indicates the air temperature at the requested location.
Visibility Range Type: single float Units: meters Values: ≥ 0	This parameter indicates the visibility range at the requested location.
Horizontal Wind Speed Type: single float Units: m/s Values: ≥ 0 Default: 0	This parameter indicates the local wind speed parallel to the ellipsoid-tangential reference plane.

Parameter	Description
Vertical Wind Speed Type: single float Units: m/s Default: 0	This parameter indicates the local vertical wind speed. Note: A positive value indicates an updraft, while a negative value indicates a downdraft.
Wind Direction Type: single float Units: degrees Values: 0.0 – 360.0 Default: 0 Datum: True North	This parameter indicates the local wind direction. Note: This is the direction from which the wind is blowing.
Barometric Pressure Type: single float Units: millibars (mb) or hectopascals (hPa) Values: ≥ 0	This parameter indicates the atmospheric pressure at the requested location.

4.2.10 Aerosol Concentration Response

The **Aerosol Concentration Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 4.1.28) 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 123 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:

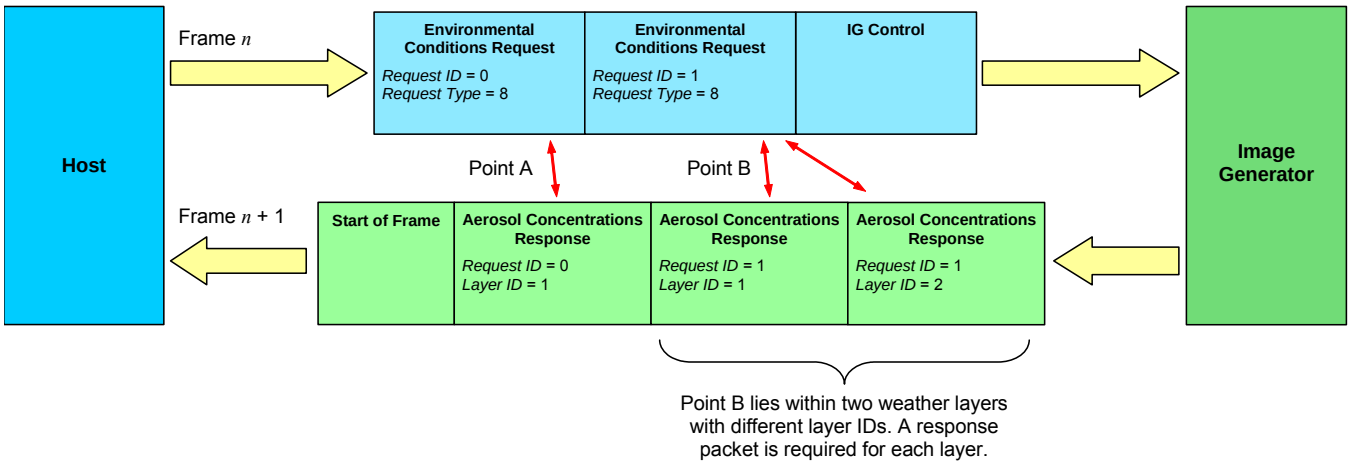


Figure 123 – Data Exchange for Aerosol Concentrations Request

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 ID = 110								*Packet Size* = 8								*Request ID*								*Layer ID*							
Aerosol Concentration																															

Figure 124 – Aerosol Concentration 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 – Aerosol Concentration Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 110	This parameter identifies this data packet as the **Aerosol Concentration Response** packet. The value of this parameter must be 110.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.
Request ID Type: unsigned int8 Units: N/A	This parameter identifies the environmental conditions request to which this response packet corresponds.
Layer ID Type: unsigned int8 Units: N/A	This parameter identifies 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	This parameter identifies the concentration of airborne particles. The type of particle is identified by the *Layer ID* parameter.

4.2.11 Maritime Surface Conditions Response

The **Maritime Surface Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 4.1.28) whose *Request Type* parameter specifies Maritime Surface Conditions. The packet describes the sea surface state at the requested geodetic latitude and longitude.

Figure 125 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:

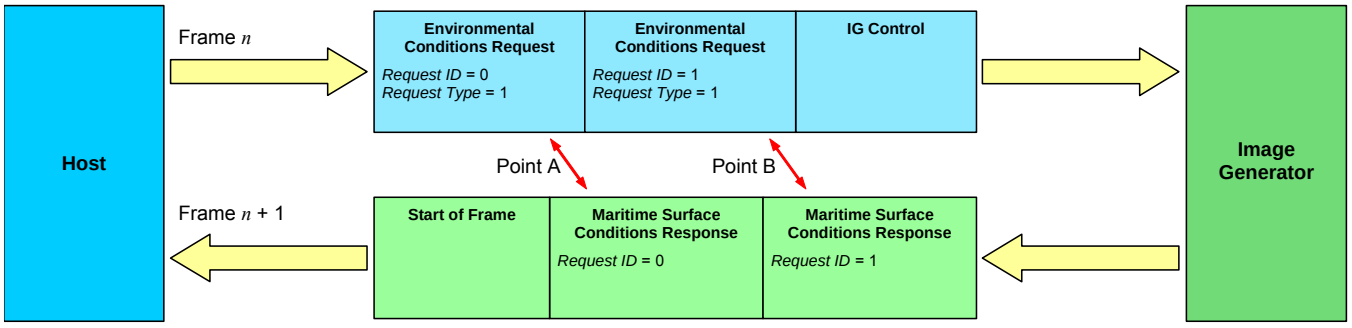


Figure 125 – Data Exchange for Maritime Surface Conditions Request

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 ID = 111	*Packet Size* = 16	*Request ID*	Reserved
Sea Surface Height			
Surface Water Temperature			
Surface Clarity			

Figure 126 – Maritime Surface Conditions 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 54 – Maritime Surface Conditions Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 111	This parameter identifies this data packet as the **Maritime Surface Conditions Response** packet. The value of this parameter must be 111.

Parameter	Description
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.
Request ID Type: unsigned int8 Units: N/A	This parameter identifies the environmental conditions request to which this response packet corresponds.
Sea Surface Height Type: single float Units: meters Datum: Mean Sea Level	This parameter indicates 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)	This parameter indicates the water temperature at the sea surface.
Surface Clarity Type: single float Units: percent Values: 0 – 100	This parameter indicates the clarity of the water at its surface. A value of 100% indicates pristine water, while a value of 0% indicates extremely turbid water.

4.2.12 Terrestrial Surface Conditions Response

The **Terrestrial Surface Conditions Response** packet is sent in response to an **Environmental Conditions Request** packet (Section 4.1.28) whose *Request Type* parameter specifies Terrestrial Surface Conditions. The packet describes the terrain surface conditions at the requested geodetic latitude and longitude.

Figure 127 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:

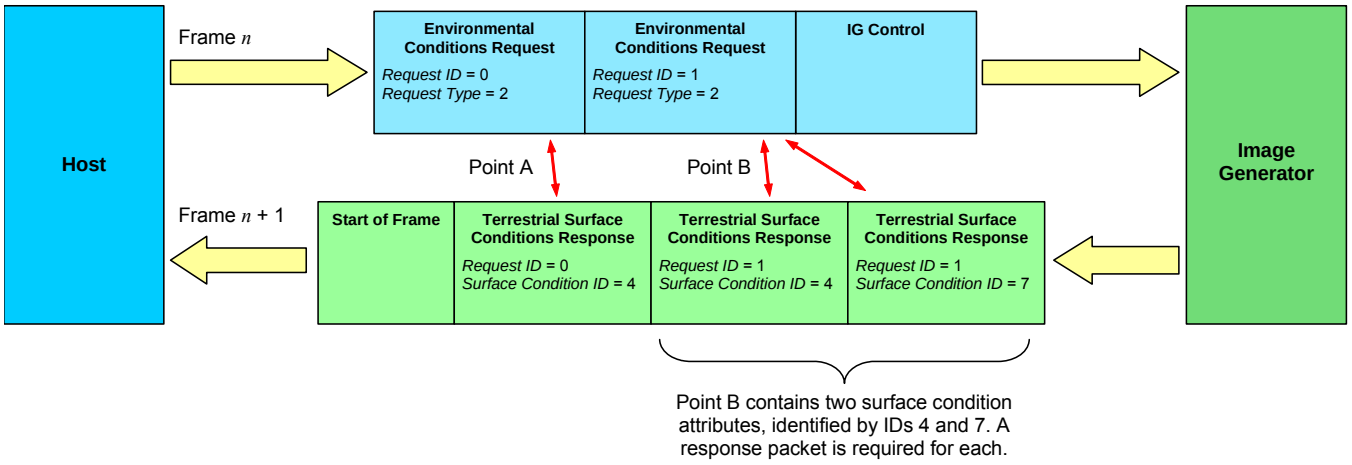


Figure 127 – Data Exchange for Terrestrial Surface Conditions Request

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 ID = 112								*Packet Size* = 8								*Request ID*								Reserved							
Surface Condition ID																															

Figure 128 – Terrestrial Surface Conditions 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 – Terrestrial Surface Conditions Response Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 112	This parameter identifies this data packet as the **Terrestrial Surface Conditions Response** packet. The value of this parameter must be 112.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.
Request ID Type: unsigned int8 Units: N/A	This parameter identifies the environmental conditions request to which this response packet corresponds.
Surface Condition ID Type: unsigned int32 Units: N/A Values: 0 – 65,535	This parameter indicates the presence of a specific surface condition or contaminant at the test point. Surface condition codes are IG-dependent.

4.2.13 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 4.1.22), 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 same the entity as the segment.

Note that collision detection testing is performed every frame by the IG. If a collision detection segment has been disabled, it will 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 ID = 113								*Packet Size* = 16								*Entity ID*															
Segment ID								Reserved							*1	*Contacted Entity ID*															
Material Code																															
Intersection Distance																															

*1 *Collision Type*

Figure 129 – Collision Detection Segment Notification 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 – Collision Detection Segment Notification Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 113	This parameter identifies this data packet as the **Collision Detection Segment Notification** packet. The value of this parameter must be 113.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity to which the collision detection segment belongs.

Parameter	Description
Segment ID Type: unsigned int8 Units: N/A	This parameter indicates 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	This parameter indicates whether the collision occurred with another entity or with a non-entity object such as the terrain.
Contacted Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity with which the collision occurred. If *Collision Type* is set to Non-entity (0), this parameter is ignored.
Material Code Type: unsigned int32 Units: N/A	This parameter indicates 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	This parameter indicates the distance along the collision test vector from the source endpoint (defined by the *X1*, *Y1*, and *Z1* parameters in the **Collision Detection Segment Definition** packet) to the point of intersection.

4.2.14 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 4.1.23). Volumes belonging to the same entity are not tested against each other.

The IG sends 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 will be sent. If a collision occurs that involves three volumes, a total of six **Collision Detection Volume Notification** packets will 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 will be excluded from all collision testing.

The contents of the **Collision Detection Volume 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 ID = 114	*Packet Size* = 16		*Entity ID*
Volume ID	Reserved	*1	*Contacted Entity ID*
Contacted Volume ID	Reserved		
Reserved			

*1 *Collision Type*

Figure 130 – Collision Detection Volume Notification 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 – Collision Detection Volume Notification Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 114	This parameter identifies this data packet as the **Collision Detection Volume Notification** packet. The value of this parameter must be 114.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.

Parameter	Description
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity to which the collision detection volume belongs.
Volume ID Type: unsigned int8 Units: N/A	This parameter indicates 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	This parameter indicates whether the collision occurred with another entity or with a non-entity object such as the terrain.
Contacted Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity with which the collision occurred. If *Collision Type* is set to Non-entity (0), this parameter is ignored.
Contacted Volume ID Type: unsigned int8 Units: N/A	This parameter indicates the ID of the collision detection volume with which the collision occurred.

4.2.15 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.

If an animation is set to play continuously, no **Animation Stop Notification** packet will be sent.

The contents of the **Animation Stop Notification** packet are as follows:

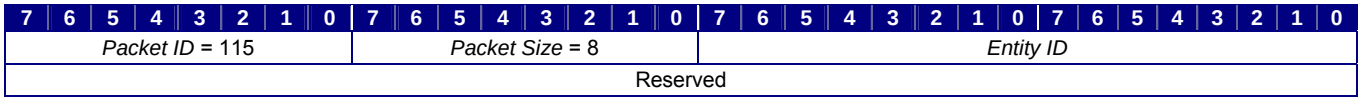


Figure 131 – Animation Stop Notification Packet Structure

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 – Animation Stop Notification Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 115	This parameter identifies this data packet as the **Animation Stop Notification** packet. The value of this parameter must be 115.
Packet Size Type: unsigned int8 Units: Bytes Value: 8	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 8.
Entity ID Type: unsigned int16 Units: N/A	This parameter indicates the entity ID of the animation that has stopped.

4.2.16 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 4.1.4) or **Short Component Control** (Section 4.1.5) packet.

This packet contains three user-defined 32-bit word values that may contain data describing the attributes of the event (e.g., time of occurrence, position). These data may be formatted as needed; however, they must be byte-swapped as 32-bit fields when byte swapping is necessary. Refer to the description of the **Component Control** packet (Section 4.1.4) 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 ID = 116								*Packet Size* = 16								*Event ID*															
Event Data 1																															
Event Data 2																															
Event Data 3																															

Figure 132 – Event Notification 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 – Event Notification Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 116	This parameter identifies this data packet as the **Event Notification** packet. The value of this parameter must be 116.
Packet Size Type: unsigned int8 Units: Bytes Value: 16	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 16.
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 will 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 will 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 will 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 will 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 will 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 will be performed correctly.

4.2.17 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 must 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 must be padded with zeroes. Zero-length messages must be terminated with four bytes containing NULL (to maintain 64-bit alignment). The maximum text length is 100 octets, including a terminating NULL.

The *Packet Size* parameter must contain the total number of bytes within the message, including the two-byte header, the message ID, the terminating NULL and any padding. This value must be an even multiple of eight (8). For example, if the string, “Error 1234,” were sent to the Host, the packet size would be 16. Figure 133 illustrates the byte allocation for the packet:

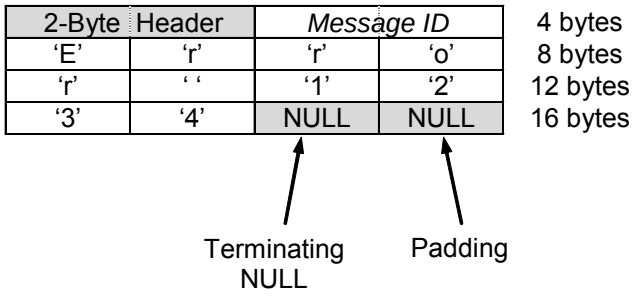


Figure 133 – Example of Image Generator Message Packet

The *Message ID* parameter identifies the message. Typically, this ID will correspond to a common, fixed message. In these cases, the Host can 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:

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 ID = 117	*Packet Size* = 4 + *message length*	*Message ID*	
Octet 1	*Octet 2*	*Octet 3*	*Octet 4*
• • •			
Octet n - 2	*Octet n - 1*	*Octet n*	NULL

Figure 134 – Image Generator Message 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 – Image Generator Message Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 117	This parameter identifies this data packet as the **Image Generation Message** packet. The value of this parameter must be 117.
Packet Size Type: unsigned int8 Units: bytes Value: 8 ≤ (4 + *message length*) ≤ 104	This parameter indicates the number of bytes in this data packet. The value of this parameter must be 4 plus the length of the message text, including NULL characters. The minimum size of the **Image Generation Message** packet is eight (8) bytes. The maximum packet size is 104 bytes. This allows for a message length of up to 100 octets, including the terminating NULL. Note: Because all packets must begin and end on a 64-bit boundary, the value of this parameter must be an even multiple of eight (8).
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.

4.3 User-Defined Packets

User-defined data packets may be used to facilitate transmission of data not specifically supported by an existing CIGI packet. User-defined data packets may be contained in both IG-to-Host and Host-to-IG messages.

When user-defined data packets are introduced into a particular CIGI application, they should adhere to the standard data packet format in order to maintain continuity across data packets. Standard data packet format includes the *Packet ID* in the first byte and the *Packet Size* in the second byte. 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 user-defined data packet depends on the amount of data contained in the packet. The developer should place the total number of bytes, including the two bytes of header, in the *Packet Size* field of this packet so that the receiver can properly interpret the packet.

Note: All user-defined packets must begin and end on a 64-bit boundary. The value of the *Packet Size* parameter must therefore be an even multiple of eight (8).

Note that when a user-defined data packet is introduced into a particular implementation of CIGI, that implementation will no longer conform to the baseline definition of the interface and thus may not be acceptable to the general CIGI user community. Ideally, each Host and IG device should be capable of being configured to ignore all user-defined packets.

The general format for user-defined packets is as follows:

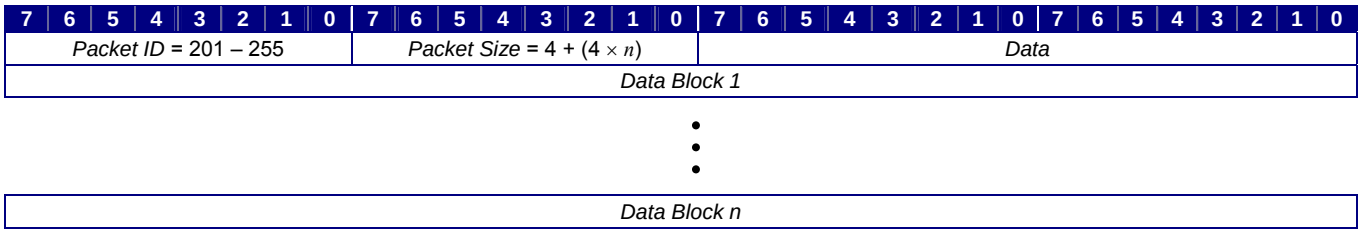


Figure 135 – General User-Defined Packet Structure

Table 61 defines each parameter's data type, units, and usage.

Table 61 – General User-Defined Packet Parameter Definitions

Parameter	Description
Packet ID Type: unsigned int8 Units: N/A Value: 201 – 255	This parameter identifies this data packet as a user-defined packet. The value of this parameter must be within the range of 201 through 255.

Parameter	Description
Packet Size Type: unsigned int8 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 must be 4 times the number of data words, plus 4 for the word that includes the two-byte header. Note: Because all packets must begin and end on a 64-bit boundary, the value of this parameter must be an even multiple of eight (8).
Data Type: Packet-specific Units: Packet-specific	This field consists of two bytes of user-defined data. The data may be arranged and formatted as needed.
Data Block n Type: Packet-specific Units: Packet-specific	These 32-bit blocks represent user-defined data. The data may be arranged and formatted as needed.

APPENDIX A – ACRONYMS

CIGI	Common Image Generator Interface
DCS	Dynamic Coordinate System
ERM	Earth Reference Model
FOV	Field of View
HAT	Height Above Terrain
HOT	Height of Terrain
IG	Image Generator
I/O	Input/Output
IP	Internet Protocol
LOS	Line of Sight
LSW	Least Significant Word
MSL	Mean Sea Level
MSW	Most Significant Word
NED	North-East-Down
OTW	Out the Window
SOF	Start of Frame
UDP	User Datagram Protocol

APPENDIX B – CIGI 3.x CHANGE HISTORY

Version 3.0

Numerous changes from CIGI 2.

Version 3.1

No data format changes from CIGI 3.0.

Version 3.2

General Changes

- Added a minor version number to the interface. Minor version changes can now contain data format changes.
- Modified the use of the Frame Counter mechanism.
- Added capabilities for continuous, periodic HAT/HOT and LOS responses from a single request.

IG Control

- Renamed *CIGI Version* parameter to “Major Version.”
- Added *Minor Version* parameter.
- Renamed *Byte Swap* parameter to “Byte Swap Magic Number.”
- Modified the use of the *Frame Counter* parameter and renamed it to “Host Frame Number.”
- Added *Last IG Frame Number* parameter.

Rate Control

- Added *Coordinate System* parameter and modified reference data for linear and angular rates.

HAT/HOT Request

- Added *Update Period* parameter.

Line of Sight Segment Request

- Added *Update Period* parameter.
- Added *Destination Entity ID* and *Destination Entity ID Valid* parameters.

Line of Sight Vector Request

- Added *Update Period* parameter.

Start of Frame

- Renamed *CIGI Version* parameter to “Major Version.”
- Added *Minor Version* parameter.

- Renamed *Byte Swap* parameter to “Byte Swap Magic Number.”
- Modified the use of the *Frame Counter* parameter and renamed it to “IG Frame Number.”
- Added *Last Host Frame Number* parameter.

HAT/HOT Response

- Added *Host Frame Number LSN* parameter.

HAT/HOT Extended Response

- Added *Host Frame Number LSN* parameter.

Line of Sight Response

- Added *Host Frame Number LSN* parameter.

Line of Sight Extended Response

- Removed *Intersection Point Coordinate System* parameter.
- Added *Host Frame Number LSN* parameter.

Sensor Response

- Renamed *Frame Counter* parameter to “Host Frame Number” and changed the use of this parameter to reflect the Host frame number.

Sensor Extended Response

- Renamed *Frame Counter* parameter to “Host Frame Number” and changed the use of this parameter to reflect the Host frame number.

Version 3.3

General Changes

- Added support for 2D symbology.
- Changed text representation from ANSI to UTF-8.

IG Control

- Added *Extrapolation/Interpolation Enable* flag to control global entity motion smoothing.

Entity Control

- Added *Linear Extrapolation/Interpolation Enable* flag to control per-entity smoothing.

Component Control

- Added “Symbol Surface” and “Symbol” component classes.

Short Component Control

- Added “Symbol Surface” and “Symbol” component classes.

Celestial Sphere Control

- Changed the time specified by the *Hour* and *Minute* parameters from local time to UTC.

Symbol Surface Definition

- New packet.

Symbol Text Definition

- New packet.

Symbol Circle Definition

- New packet.

Symbol Line Definition

- New packet.

Symbol Clone

- New packet.

Symbol Control

- New packet.

Short Symbol Control

- New packet.

APPENDIX C – ERRATA

This appendix contains corrections to this ICD. Changes and additions are indicated by highlighted text. Deletions are indicated by ~~struckthrough text~~. These pages supercede the corresponding sections within the document. Changes included here will be incorporated into the next revision of the ICD.

This appendix will be updated periodically. Visit the CIGI web site (http://cigi.sourceforge.net) for the latest version.

The list below gives the sections and page numbers that contain the corrections, describes each change, and provides the revision date for each change.

Revision Date	Section	Page	Page(s) Affected	Description