TA Document 2001001
AV/C Panel Subunit Specification 1.1

May 8, 2001

Sponsored by:
1394 Trade Association

Accepted for Release by:
1394 Trade Association Board of Directors.

Abstract:
This specification defines a model and command set for a Panel subunit operating over the Function Control Protocol defined by IEC 61883. The Panel subunit provides device GUI information for remote control. The mechanism for user commands and GUI data transmission is defined.

Keywords:
Audio, Video, 1394, Digital, Interface, GUI, OSD, on screen display, Panel, Menu, Remote control.

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in working groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
Regency Plaza Suite 350
2350 Mission College Blvd.
Santa Clara, Calif. 95054, USA

Table of Contents

List of Figures

List of Tables

1. Overview

1.1 Purpose

This document specifies the AV/C Panel subunit. The Panel subunit provides user interface (UI) services including on-screen display (OSD), graphical user interface (GUI) services for AV/C devices. It provides a model for describing and accessing the physical or logical controls on a device from a remote location. The Panel subunit defines several data structures that describe standardized control types (such as buttons, text entry fields, etc.), and a user-manipulation command set for accessing these controls. A controller that wishes to represent a target device to the user gets the GUI data from the Panel subunit and constructs a user interface. The user manipulates these on-screen representations or operates a button on a physical or logical remote commander of the controller, and then the controller sends commands back to the panel subunit. In response to these commands, the panel subunit performs some action(s) and updates the GUI if necessary. And this specification is updated for adding a few *operation_id* of PASS THROUGH command in order to realize new key.

1.2 Scope

The Panel subunit is a user-centric model; the user decides which controls to use and when to use them. In addition, it allows the target to display spontaneous messages to the user, in situations where this may be necessary.

The Panel subunit allows the user manipulations to be transferred to the target device when the user manipulates the GUI representation on the controller's screen or operates a button on the physical or logical remote commander of the controller.

This model allows the total encapsulation of knowledge about the device state and state transition rules inside the target. By making use of status inquiry and state change notification services, any controller can properly display the current state of the target at any time. Thus, it is not necessary for the controller to know anything about what happens when a particular control is manipulated; it simply renders the GUI elements in their current states as indicated by the panel subunit.

This model also allows the controller to get user interface information from target device by using another way, e.g. stream data, EIA-775 manner, which is not described in this specification. Even in this case, the Panel subunit provides a user manipulation mechanism for target device.

In version 1.1 of this specification, additional *operation_id*s of PASS THROUGH command are defined as enhancement of version 1.0. Only necessary parts for this change are updated in version 1.1, and other discussion items regarding Panel subunit enhancement are out of scope.

2. References

The following standards contain provisions, which through reference in this document, constitute provisions of this standard. All the standards listed are normative references. Informative references are given in Annex A. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

2.1 1394 Trade Association Specifications

[R1] AV/C Digital Interface Command Set General Specification, Version 3.0. TA document number 1998003.

[R2] Enhancements to the AV/C General Specification Version 3.0, Version 1.0. TA document number 1998010.

[R3] AV/C commands for management of Asynchronous Serial Bus Connections Version 1.0. TA document number 1998011.

2.2 Related Technical Specifications

[R4] IEEE Std 1394-1995, Standard for a High Performance Serial Bus

[R5] IEEE Std 1394a-2000, Standard for a High Performance Serial Bus - Amendment

[R6] IEC 61883, Consumer audio/video equipment – Digital interface – Part 1: General.

[R7] ISO/IEC 13213:1994, [ANSI/IEEE Std 1212, 1994 Edition], Information Technology— Microprocessor systems— Control and Status Register (CSR) Architecture for Microcomputer Buses.

[R8] UNICODE Standard Version 2.0, http://www.unicode.org

[R9] IETF RFC2083, W3C, Portable Network Graphics (PNG) specification Version 1.0

[R10] Audio Interchange File Format, Version C, allowing for Compression(AIFF-C), Digital Audio Visual Council (DAVIC) 1.3 Part 9, Annex B

[R11] EIA-775 DTV 1394 Interface Specification

[R12] HAVi version 1.0 specification, http://www.havi.org

[R13] ISO/IEC 13818-1:1996(E), Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Systems

[R14] IEEE1212-2000, Standard for a Control and Status Registers (CSR) Architecture for microcomputer buses.

[R15] ISO/IEC 10646-1:1993, Information technology - Universal Multiple-Octet Coded Character Set (UCS)

[R16] ETS 300 706, Enhanced Teletext Specification

[R17] ARIB TR-B15, BS Digital Broadcast Operation Specification. [Japanese Only]

3. Definitions

3.1 Conformance Levels

3.1.1 expected: A key word used to describe the behavior of the hardware or software in the design models *assumed* by this Specification. Other hardware and software design models may also be implemented.

3.1.2 may: A key word that indicates flexibility of choice with *no implied preference*.

3.1.3 shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

3.1.4 should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

3.1.5 reserved fields: A set of bits within a data structure that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall zero these fields. Future revisions of this specification, however, may define their usage.

3.1.6 reserved values: A set of values for a field that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall not generate these values for the field. Future revisions of this specification, however, may define their usage.

The IEEE is investigating whether the “may, shall, should” and possibly “expected” terms will be formally defined by IEEE. If and when this occurs, draft editors should obtain their conformance definitions from the latest IEEE style document.

3.2 Glossary of Terms

3.2.1 byte: Eight bits of data, used as a synonym for octet.

3.2.2 CSR Architecture: A convenient abbreviation of the following reference[R7] : ISO/IEC 13213 : 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems— Control and Status Register (CSR) Architecture for Microcomputer Buses.

3.2.3 quadlet: Four bytes of data.

3.2.4 AV unit: The physical instantiation of a consumer electronic device, *e.g.*, a camcorder or a VCR, within a Serial Bus node. This document describes a command set that is part of the software unit architecture for AV units.

3.2.5 AV subunit: an instantiation of a virtual entity that can be identified uniquely within an AV unit and offers a set of coherent functions.

3.2.6 node: An addressable device attached to Serial Bus with at least the minimum set of control registers defined by IEEE Std 1394–1995.

3.2.7 node ID: A 16-bit number, unique within the context of an interconnected group of Serial Buses. The node ID is used to identify both the source and destination of Serial Bus asynchronous data packets. It can identify one single device within the addressable group of Serial Buses (unicast), or it can identify all devices (broadcast).

3.2.8 PCR: Plug Control Register, as defined by IEC 61883 [R6], Digital Interface for Consumer Electronic Audio/Video Equipment.

3.2.9 iPCR: Input plug PCR, as defined by IEC 61883 [R6].

3.2.10 oPCR: Output plug PCR, as defined by IEC 61883 [R6].

3.2.11 plug: A physical or virtual end-point of connection implemented by an AV unit or subunit that may receive or transmit isochronous or other data. Plugs may be Serial Bus plugs, accessible through the PCR's; they may be external, physical plugs on the AV unit; or they may be internal virtual plugs implemented by the AV subunits.

3.2.12 source plug: A subunit source plug is a source of output data

3.2.13 GUI: Graphical User Interface - refers to the user interface of a target device which is presented on a controlling device using graphics and/or text. Normally the controller would be a television or PC screen, but it could be *any* device with some kind of display capability.

3.2.14 Element: An element is an item used in the GUI, i.e. Panel, Group or GUI element.

3.2.15 Panel: A panel is a collection of GUI elements. The GUI elements may be independent of each other, or they may be collected into groups within the panel. The panel is sometimes also called a Control Panel, since it handles GUI elements which control the target device.

3.2.16 Group: A group is a logical collection of GUI elements within a panel. This mechanism forms a simple hierarchy in which a panel may contain groups, and the groups contain individual GUI elements. Groups are used by controllers to present parts of a panel, when the controller does not have the ability to display the full panel at one time.

3.2.17 GUI Element: Each item in the GUI is a GUI element, such as a button, text entry field, icon, etc. Some GUI elements are interactive, and others are for display only.

3.2.18 Data entity: A data entity is an item that contains raw data for GUI, such as text, audio, bitmap.

3.2.19 Menu: This is another term for a panel, or collection of GUI elements. The traditional VCR-programming model often uses the term Menu to describe the collection of information which is displayed to the user at any given time.

3.2.20 OSD: On-Screen Display - this refers to the graphical user interface of a target device which is presented on a controlling device. Normally this would be a display of the graphics or text.

3.2.21 root panel: A root panel is a panel which is first displayed to the user for providing guide information for the interaction/control, e.g. it has typically functions as a main menu of the subunit. The root panel may be content menu, setup menu or the other kind of menu, and it can be changed according to the target status.

3.2.22 Stream data: A time-ordered digital data e.g. real-time audio or video data.

3.2.23 Safety display area: An area is intended to display on screen all the GUI as same look as it is designed. .

3.3 Acronyms and Abbreviations

AC Asynchronous connection

AC-CTR Asynchronous connection controller

AV/C Audio/Video Control

CSR A node or unit Control and Status Register, as defined in ISO/IEC 13213:1994 [R7]

EUI-64 Extended Unique Identifier, 64-bits, as defined by the IEEE. The EUI-64 is a concatenation of the 24-bit company_ID obtained from the IEEE Registration Authority Committee (RAC) and a 40-bit number (typically a silicon serial number) that the vendor identified by company_ID guarantees to be unique for all of its products. The EUI-64 is also known as the node unique ID and is redundantly present in a node's configuration ROM in both the Bus_Info_Block and the Node_Unique_Id leaf.Extended Unique Identifier, 64 bits, as defined by IEEE

FCP Function Control Protocol, as defined by IEC 61883 [R6], Digital Interface for Consumer Electronic Audio/Video Equipment.

IEEE The Institute of Electrical and Electronics Engineers, Inc.

PS-CTR Panel subunit controller

4. The Panel Subunit Model

4.1 The AV/C Panel Subunit Model

The Panel subunit provides user interface (UI) services for AV/C devices. It provides a model for describing and accessing the physical or logical controls on a device from a remote location. The Panel subunit defines several data structures that describe standardized control types, and a set of user-manipulation commands for accessing these controls.

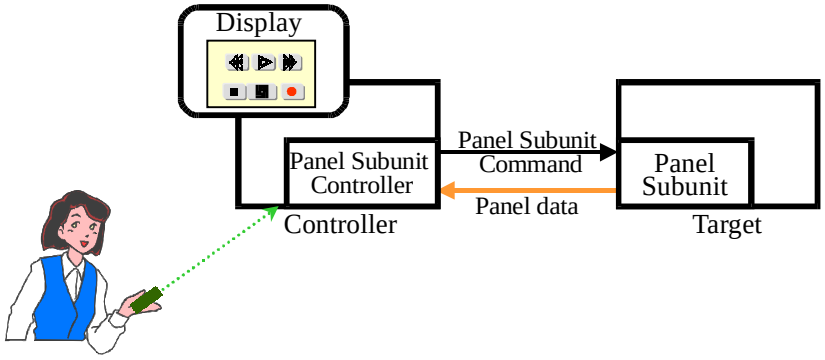

Figure 4.1 – Panel subunit model

The panel subunit model allows controllers to get the UI data from the Panel subunit and construct a user interface on the controller's screen. The asynchronous connection or other mechanisms, e.g. data streaming, are used to get UI information.

The panel subunit controller (PS-CTR) is used to control the panel subunit according to the user operation using certain controller dependent manners, e.g. IR remote commander. The user manipulates the user interface on the controller's screen or operates a button on a physical or logical remote commander of the controller, and then the controller sends commands back to the panel subunit. In response to these commands, the panel subunit performs some action(s) and updates the UI if necessary.

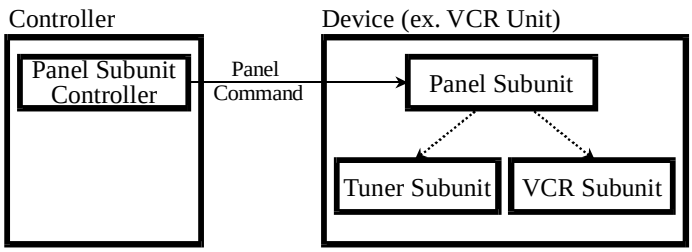

Figure 4.2 – System model using Panel subunit

Even though there may be several panels, or menus, in a target device, the target device shall have only one panel subunit.

Unlike many other AV/C subunits, the panel subunit does not directly deal with media streams itself. The main purpose to use a panel subunit is to allow it to translate the incoming user action commands into internal actions, which affect other subunits and/or the unit, and dispatch them to appropriate subunit or unit inside the target device using target device dependent manner. The result of these actions *may* have an effect on media streams.

4.2 Panel Subunit Logical Model

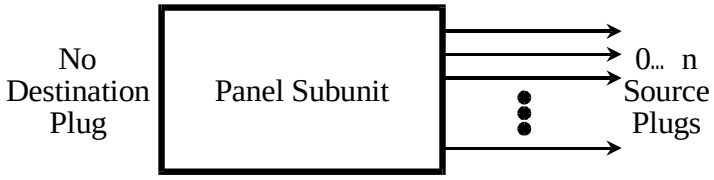

Figure 4.3 – Panel subunit logical model

The panel subunit may support multiple asynchronous outputs, used for GUI data transmission, depending on implementation. In case of the direct mode, where the panel subunit itself transfers GUI data, the panel subunit has at least one source plug. In case of the indirect mode, where the panel subunit doesn't care about GUI data, the panel subunit has no source plug.

4.3 UI data transfer mode

A Panel subunit has two UI data transfer modes.

1) **Direct mode:** allows Panel subunit to transfer UI data and to receive user manipulations. The controller constructs UI from the obtained UI data, and conveys user operations to a target by mainly USER ACTION commands, and sometimes PASS THROUGH commands. For details, refer to section 4.4.

2) **Indirect mode:** allows Panel subunit to receive user manipulations but the panel subunit doesn't care about UI data to be displayed on the controller screen. The GUI image might be transmitted to the display device by being multiplexed with the stream data transmitted from other subunit than the panel subunit in the target device. The Panel subunit controller conveys user operations to the target by only PASS THROUGH commands, and works simply like a remote commander of the target device. For details, refer to section 4.5

A target shall be implemented with only one of the above UI data transfer modes.

The controller can identify which mode the panel subunit supports, by using the following procedure;

1) **Direct mode:** The panel subunit supports GUI UPDATE status or control command. If the controller wants to check the capability of the panel subunit, it issues the GUI UPDATE status command. And the response is NOT "NOT IMPLEMENTED" response, then the panel subunit supports the direct mode. The controller can also check the GUI UPDATE control command, skipping the status command, if it wants to check the mandatory display capability, e.g. 640 x 480 pixels GUI resolution.

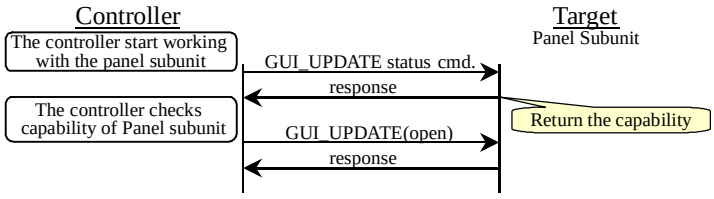

Figure 4.4 – Start procedure for direct mode

2) **Indirect mode:** The panel subunit doesn't support GUI UPDATE status nor control command. First, the controller issues the GUI UPDATE status command. Then if the response is "NOT IMPLEMENTED", the panel subunit doesn't support direct mode. So the controller will check whether it supports the indirect mode by using GENERAL INQUIRY command for PASS THROUGH command. If the response is "IMPLEMENTED", the panel subunit supports the indirect mode.

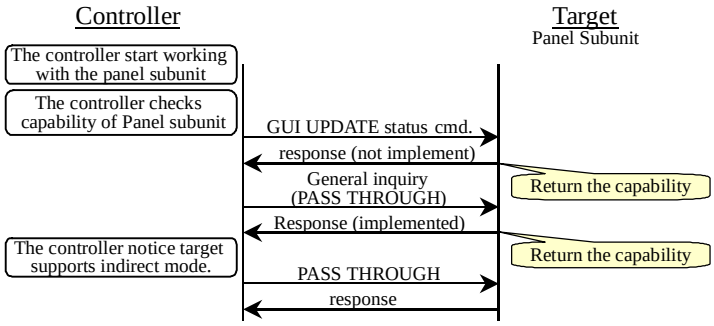

Figure 4.5 – Start procedure for indirect mode

4.4 Direct mode

In the direct mode, the panel subunit provides the UI data from its source plug.

The direct mode allows controllers to get the UI data from the Panel subunit using asynchronous connection to, construct a user interface. In this case the controller works as an asynchronous connection controller (AC-CTR) as well as a panel subunit controller (PS-CTR). AC-CTR is used to establish an asynchronous connection via some connection procedure between the target (containing the panel subunit) and the controller that has the capability of rendering and displaying UI.

The user manipulates the user interface on the controller's screen or operates a button on a physical or logical remote commander of the controller, and then the controller sends USER ACTION or PASS THROUGH commands back to the panel subunit. In response to these commands, the panel subunit performs some action(s) and sends UI updates to the controller if needed.

Even though there may be several panels, or menus, in a target device, the device has only one panel subunit and the UI data structures allow this single subunit to represent and manage any number of separate control panels for its device. A single control panel mayyy have sets of GUI elements which trigger actions in different subunits and/or the unit. For example, a panel for a VCR may have the typical Stop, Play and Record buttons which affect the VCR subunit, and it may also have a set of preset buttons which affect the tuner subunit. It is also possible that the control panel has GUI elements which don't affect any other subunits; they cause a change in the state of the unit. A text entry field which allows the user to store a preferred name for the device would be such an example.

The collection of GUI elements which appear in a panel is determined by the designer of the panel; there are no restrictions on this collection other than the product specifications and the common sense of user interface design.

In this mode the panel subunit can have only one controller for each source plug, and each source plug works according to commands of the owner associated with that source plug.

4.4.1 GUI Layout and Presentation for direct mode

The layout rules of GUI elements are based on geometric coordinates and use relative x, y values for each GUI element. GUI elements are arranged in a hierarchy and positioned relative to their parents. The parents of individual GUI elements are either groups or panels; the parent of a group is the panel in which it is defined. Panels have no parent – their placement on screen is determined by the controller.

The following diagram illustrates this relationship:

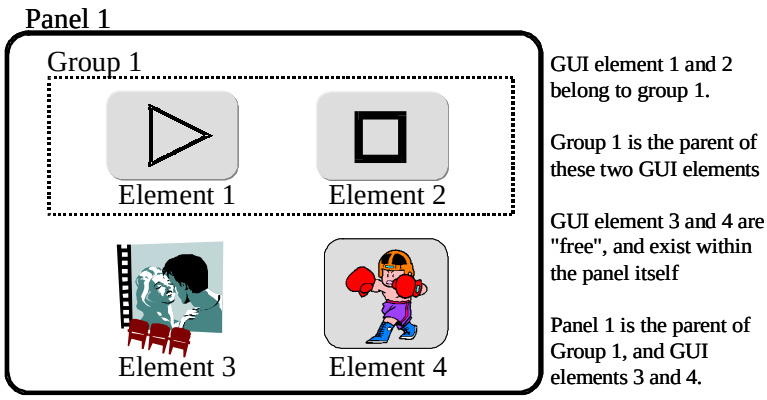

Figure 4.6 – Relationship among Panel, Group and GUI elements

The panel subunit *suggests* a preferred layout, which is encoded into the GUI data structures. However, the controller may tailor the presentation of GUI elements based on its own limitations, such as the screen size, the ability to display graphics or text only, etc.

This mechanism provides a very flexible layout and display model, from text-only through high quality graphics presentation. As a result, as long as there is some kind of display available to the controller, this model can guarantee some kind of presentation to the user.

However, the controller cannot provide guarantees over the graphical rendition of GUI elements, since their representation may be changed due to lack of display screen space, color and transparency capabilities, or other resource limitations. The controller may discard some data or elements right after they are received.

Furthermore, application software can create different representations of the GUI, using the GUI elements as “hints”. For instance the application software can display quite deferent look and feel using icons that is fetched from the panel subunit. However the controller should try to preserve the appearance of GUI elements as much as it can, based on its rendering capabilities.

4.4.2 Interactions between controller and target

There are two interactions: one between the user and the controller, and the other between the controller and the target.

The controller communicates with the user via the input and output devices of (typically) the device upon which the controller is executing. This I/O communication may be done in a controller-implementation-dependent manner (e.g. IR remote, keyboard and mouse, television or PC display, small text-only screen, etc.). The target controls itself in a target-implementation-dependent manner. The protocol described in this section is structured so that a controller may be defined in a generic manner. That is, a controller does *not* have to be implemented with knowledge of a particular target in mind; *all* target-dependencies are represented in the GUI data provided by the target to the controller.

It should be emphasized that the controller usually does not understand what happens as a result of issuing a user input control message to the target. So in addition to well-known functionalities, it is possible to handle new functionality on the target device which cannot be assumed in advance.

The direct mode uses an asymmetric data transfer (ADT) model for the controller-target interaction. With this ADT model, the control data from controller to target is transferred using AV/C commands, while the GUI data from target to controller is transferred using an Asynchronous Connection.

The following diagram illustrates a typical interaction sequence between a controller and a target in the direct mode:

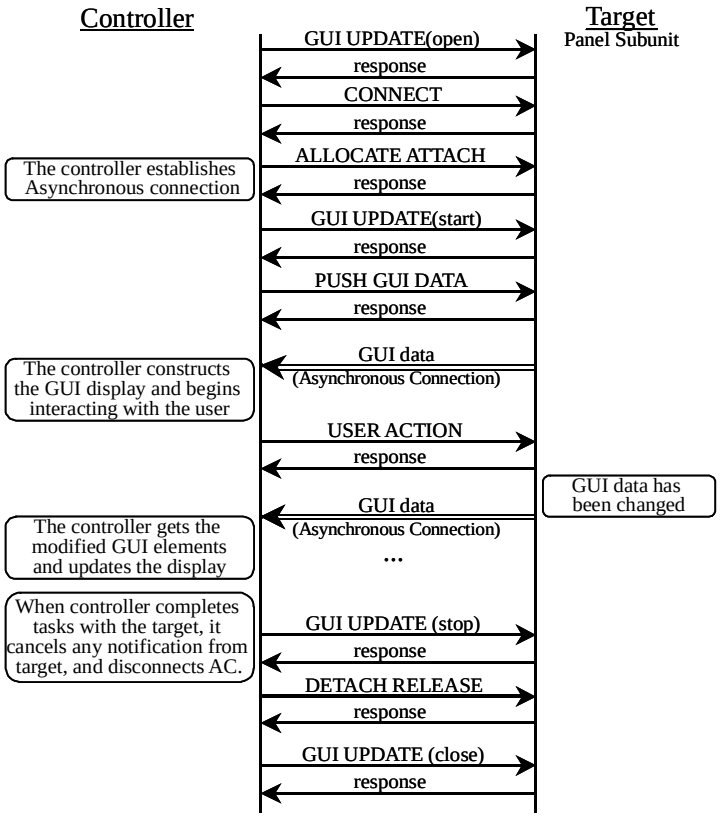

Figure 4.7 – Detailed Interactions between Controller and Panel subunit

When the controller starts working with the panel subunit, the controller will first check the availability of a source plug of the panel subunit and reserve the source plug using GUI UPDATE control command with open subfunction. If the command is accepted, the controller gets the ownership of the source plug.

Then the controller makes the internal connection between the target unit and the panel subunit by using CONNECT command specified in reference [R1].

After that, the controller makes an asynchronous connection between the controller unit and the unit that contains the panel subunit, using ALLOCATE ATTACH subfunction in ASYNCHRONOUS CONNECTION command, etc. specified in the reference [R3]. If all connections succeed, the data path between the controller and the panel subunit is established.

Note that the controller may use different way to establish the asynchronous connection if the other ways are specified. However the connection between the controller and the panel subunit shall be established

after GUI UPDATE with open subfunction is accepted and before the controller issues GUI UPDATE with start subfunction.

The communication process starts with the controller sending the GUI UPDATE command with “start” subfunction to the panel subunit to interact with the panel subunit. This command also indicates the notification scope: a full or partial panel. The notification scope can be changed any time by issuing a new GUI UPDATE command with “change” subfunction.

The PUSH GUI DATA command is a request for some specific GUI element(s). The target sends the requested GUI information over the asynchronous connection. The controller starts accessing some or all of the GUI data structures (panels, groups or GUI elements). The controller does this by first sending the GUI request, i.e. PUSH GUI DATA command, to get the root panel, i.e. the first menu of the target, or some GUI elements with the root panel. After getting the GUI data, the controller renders the GUI elements on the display screen, and begins interacting with the user. Thereafter, the controller may:

1) update the GUI representation that is being displayed to the user by a controller implementation-dependent mechanism;
2) ask the target for more GUI elements using the PUSH GUI DATA; or
3) send user action information to the target according to the contents of the GUI element and the user input.

At any time after a controller has requested notification from the target, using the start subfunction of the GUI UPDATE command, and before the controller cancels the notification, using the stop subfunction of the GUI UPDATE command, the target may indicate the changed GUI element(s) to the controller. For example, indicating that some aspect of the target’s internal state which is relevant to the user has changed. The indication will be done by the target spontaneously sending GUI elements to the controller over the asynchronous connection and will provide the controller with information about which parts of the target’s GUI presentation have changed.

When the user operates the GUI element, the controller sends the user action by USER ACTION control command to the panel subunit, and the USER ACTION control command is generally accepted. If one or more GUI elements have been changed as a result of the user action, the panel subunit sends the changed GUI element(s) to the controller over the asynchronous connection.

For example, a GUI button pressed by the user may attempt to place a target VCR into a “rewinding” mode and the target may wish to indicate the success (or failure) of the command by sending a TEXT element with an appropriate string value over asynchronous connection.

In this manner the user and the target device communicate with each other through a sequence of commands and responses sent between the controller and the target, guided by the GUI data the controller obtained from the target panel subunit.

When the controller stops interacting with a target, it sends a GUI UPDATE command with “stop” subfunction to that target; this cancels ALL outstanding notification requests for that controller. The target shall thereafter not send any changed GUI data to that controller and the controller shall not send any other USER ACTION commands to that target. The controller may restart the interaction by sending another GUI UPDATE command with “start” subfunction to the same target.

If there are no more interactions between the controller and the target and after “stop” subfunction is accepted, the AC-CTR can issue DETACH_RELEASE subfunction in ASYNCHRONOUS CONNECTION command to the target to close the asynchronous connection.

After the asynchronous connection between the controller and the target is broken, the controller sends a GUI UPDATE command with “close” subfunction to that target; this cancels the controller’s ownership for

the specified source plug. The target may accept GUI UPDATE command from any controller and assign new ownership for the source plug. The controller may open a new interaction by sending another GUI UPDATE command with “open” subfunction to that target.

The controller may send the PASS THROUGH control command except for cursor navigation, e.g. right, left, up and down, and select operations anytime. However it is target dependent whether or not the target works according to the specified operation, e.g. target device might not react to it except for sending back NOT IMPLEMENTED or ACCEPTED response. If the controller wishes to send PASS THROUGH control commands for menu operations, e.g. root menu or setup menu, it should establish connection between the target and itself and complete start procedure for direct mode. Thereafter the controller may send PASS THROUGH control command for menu operation. The PASS THROUGH commands such as “root menu” or “play” operation may be used for a short-cut manipulation in the direct mode.

The PASS THROUGH control command for cursor navigation, i.e. right, left, up and down, and select operations is valid only when the cursor through attribute is enabled in the rendered GUI element. For more details of the PASS THROUGH command, refer to section 9.

4.4.3 User Output and Input Device Models

The controller manages in an implementation-dependent manner a display device to communicate to the user and a user input device to accept commands from the user. In order to allow the model to apply to a broad range of particular devices, these devices are modeled abstractly by only specifying in a general way in which GUI element data is associated with physical user interaction. The data and its constituent GUI elements provide “suggestions” on how the controller is to make this association. These suggestions depend on the type and attribute values of the GUI elements.

4.4.3.1 Output Device Model

The GUI elements that the controller obtains from the panel subunit may be physically presented, *rendered*, to the user using the display device. Each GUI element is of a particular type (e.g., panel, icon, button) and each type of GUI element has a particular set of attributes (e.g., size, position, color, image content, sound content). All attributes are divided into two distinct classes: mandatory attributes and optional attributes. A GUI element’s mandatory attributes will always have an associated value; an optional attribute may or may not have an associated value. Every GUI element type has one or more mandatory *label* attributes whose values are text strings.

There are three broad classes in the panel subunit controller:

Full Capability - all GUI elements are exactly rendered as described at the screen size point of view, i.e. screen size is bigger than or equal to the GUI resolution, which is the size of safety display area for GUI data. The minimum GUI resolution size is 640 x 480 pixels.

Intermediate – the entire GUI elements cannot be displayed as given, e.g. the screen size is smaller than the GUI resolution size and/or some attributes, e.g. sound data, may not be considered.

Basic – Only a few or even a single GUI element can be displayed at a time and no other attributes are supported even if they are mandatory. However, at least, its text string label shall be rendered if its label is not empty.

Note that the controller is also responsible for physically presenting the values of sound attributes of GUI elements as best as it and its display device are able to. The controller may choose one of three display capabilities according to its situation, e.g. plural panels are displayed on the same screen at a time.

Many GUI element types have attributes that are to be used by the controller to determine position and size on the display device. The physical display device is considered to be a rectangular array of discrete pixels. Position and size information is expressed with respect to a two-dimensional coordinate system for this rectangle with non-negative x and y coordinate values; the upper-left corner as the user faces the device is <0,0>. The positions of GUI elements contained within *organizational* GUI elements (that is, panel and group elements) are relative to the position of the most immediately containing organizational element.

For many types of GUI elements, attributes that specify their position are optional. When a position attribute is not given for an element the controller has broad freedom to locate the rendering of that element subject only to the guidelines provided by the place of that element within the overall GUI element data – see section 4.4.4.1 Organizational GUI Elements.

4.4.3.2 Input Device Model

Like output to the user, input from the user is modeled abstractly. In order to present this model a number of definitions are made. GUI element types are ***interactive*** (e.g., button, set range) or ***non-interactive*** (e.g., status), based upon whether they, respectively, can or cannot be used by the controller to send a USER ACTION message to the target. GUI element types are ***user-modifiable*** (e.g. set range) or ***non-user-modifiable*** (e.g., button, icon) depending upon whether they, respectively, have or do not have a user-modifiable attribute.

User input is taken into account within the model by requiring that a controller, in a manner appropriate for the actual physical input device, allow the user to:

1) Change the value of a user-modifiable attribute of a user-modifiable GUI element (e.g., enter a new text string into a GUI entry element).
2) Select an interactive GUI element (e.g., “press” a GUI button element) — this causes the controller to send a USER ACTION message to the target with arguments whose values depend on the type of the GUI element and on the particular kind of selection the user performed.
3) Explicitly present on the display device another panel by selecting a Panel link element. This only affects the display device; it does NOT directly cause the controller to send a USER ACTION command to the target. This may, however, cause the controller to obtain more GUI elements from the target using PUSH GUI DATA command.
4) Change the display by having the controller render GUI elements that are associated with the current panel but which are not currently rendered. This change only affects the display device; it does not directly cause the controller to send a USER ACTION message to the target. Note that if display resources are otherwise inadequate to render the new GUI elements, the controller may “un-render” GUI elements in a manner of its own choosing. For example, the controller may remove an arbitrary element from the current display or allow the user to control a scrolling mechanism. The scrolling mechanism allows the user to bring in or remove GUI elements as desired. Note that scrolling does not necessarily mean scrollbar-type scrolling; a page-flipping mechanism is another example of scrolling.

4.4.4 GUI Elements

4.4.4.1 Organizational GUI Elements

The GUI elements contained in a target’s data are arranged into a hierarchy. This hierarchy serves three purposes:

1) It allows a controller to navigate through the GUI elements in an "organized” way.

2) From the target's point of view, it indicates which GUI elements belong logically together and should, therefore, preferably be displayed physically together to the user.

3) From the controller's point of view, it can be used to let the target know about which GUI element changes the controller should be notified.

Two types of *organizational* GUI elements determine the hierarchical organization of the elements in a target's data:

1) The *panel* element, which is a list of the GUI elements that are contained in the panel. Panel elements may *not* be (directly) contained in other panel elements.

2) The *group* element, which is a list of the GUI elements that are contained in the group. Group elements may *not* be (directly) contained in other group elements.

Panel and group elements are non-interactive and non-user-modifiable in the sense defined in section 4.4.3.2 Input Device Model.

Organizational GUI elements are represented as list of GUI elements.

A *non-organizational* GUI element is defined to be an element that is not an organizational GUI element. Non-organizational GUI elements are written in section 4.4.4.2 Non-Organizational GUI Elements.

Non-organizational GUI elements are represented as list of data entities, contained within lists.

At any point in time a controller has a *current panel* which is the panel element that the controller has most recently obtained from the target, and which the controller is currently rendering as best it can (with all its contained GUI elements) on the display device.

The controller allows the user to navigate through the current panel's contents by means of the above hierarchy. The controller can choose to render panels, groups within panels and non-organizational GUI elements within panels or groups. The way in which a controller navigates through the panels and groups is its choice, although it is possible for the target to specify *suggestions* for navigation. The only thing that a target can assume is that the user is aware of the panel and the group that contains any GUI element the user may select. The controller (depending on the capabilities of its display device) can assure this in a number of ways:

1) by displaying the complete panel, and all its groups with all their non-organizational GUI elements

2) by displaying for the panels and groups only their label or icon

3) by displaying the non-organizational GUI elements only (in this case, the controller can still assure that the user knows what he is looking at, because he steered the navigation through the panels, groups, and GUI elements in a particular "known to the user" way)

4) some combination of the above

4.4.4.1.1 Uses of Organizational GUI Elements

A **panel** (and its contained elements) may be used for the presentation and control of a function or a very closely related set of functions in the target device. The panel represents a set of GUI elements which the controller should render together on a single display screen. If this is not possible, the controller may divide/modify the set of panel elements over as few display screens as display capability allows. However, this must be done in a manner consistent with the intention of the designer of the target GUI element data, so that the user may easily think of this set of elements as comprising a whole.

Similarly, a **group** may be used for the presentation and control of a sub-function of the target device. The elements contained in the group all have the same level of display priority. In situations where the controller cannot render all the groups and other elements in a panel at the exact positions specified by their attributes, the controller may move or choose to (temporarily) not render some groups or other elements. However, the controller must make a strong attempt to keep the elements in a group together when they are rendered. The groups within a panel have a linear ordering of their priorities. The controller, when deciding where to position groups and their contained elements or which groups to temporarily not render, should favor groups of higher priority.

Like the panel and group elements which may contain them, non-organizational GUI elements have position information supplied by their attributes. If display resources are limited, the controller may change the position of non-organizational GUI elements within their panel or group.

4.4.4.2 Non-Organizational GUI Elements

As defined in the section above, non-organizational GUI elements are those GUI elements that are not panel or group elements. These types of GUI elements occupy "leaf" positions in the GUI hierarchy determined by panel and group elements. A non-organizational GUI element has mandatory and optional attributes that suggest to the controller:

1) how the element should be rendered
2) what sort of effect user input should have upon the element; i.e., is the element *user-modifiable* in the sense defined in section 4.4.3.2 Input Device Model
3) what effect user selection should have with respect to the controller sending USER ACTION messages to the target; i.e., is the element *interactive* in the sense defined in section 4.4.3.2 Input Device Model

The following list briefly describes a subset of non-organizational GUI elements. For each element type, information includes its important mandatory attributes, its use and typical renderings, and categorizing it with respect to interactivity and user-modifiability. Detailed information is provided in section 8.1.2.

1) **TEXT** – has a mandatory attribute containing a text string. This element is used to present a static label or other textual information to the user. It may be interactive and non-user-modifiable.
2) **ICON**– has mandatory attributes containing a reference to an image bitmap. Used to allow a user to send a simple (i.e., without parameters) command to the controller if this is interactive. It may be interactive and non-user-modifiable.
3) **BUTTON** – has mandatory attributes that describe a sequence of "pressed" and "released" appearances. Used to allow a user to send a simple (i.e., without parameters) command to the controller. It may be interactive and non-user-modifiable.
4) **ANIMATION** – has a mandatory attribute containing a sequence of icons. If there is only one icon in the sequence, that icon is statically rendered by the controller. If there is more than one icon in the sequence, the controller renders each icon in temporal sequence giving the user the effect of an animation. It may be interactive and non-user-modifiable.
5) **CHOICE** – has mandatory attributes that describe a discrete set of possible values of which the user may choose one or several, and thereby indicate to the target some command or course of action. Typical renderings are as many-out-of-many "check boxes" or as one-out-of-many "radio buttons". It may be interactive and user-modifiable.
6) **ENTRY** – allows a user to enter a text string value and send it to the target. A typical rendering is as a text entry field perhaps with an associated on-screen keyboard. It may be interactive and user-modifiable.

7) **SHOW RANGE** – has mandatory attributes defining a numeric range and a particular value within that range. Used to present dynamic numeric information to the user. Typical renderings are as a circular meter with variable position pointer or as a linear variable length bar. It may non-interactive and non-user-modifiable.

8) **SET RANGE**– has mandatory attributes defining a numeric range. Used to allow a user to send a command with a numeric parameter to the target. Typical renderings are as a slider or dial. It may be interactive and user-modifiable.

9) **PANEL LINK** – has a mandatory attribute containing the name of a panel element. Used for user-driven navigation among Panels by the controller. See sections 4.4.5 and 10.5.3. It may be non-interactive and non-user-modifiable.

Again, the actual appearance and position of each GUI element depends on the controller; while a target's data provides an explicit logical structure for its GUI elements, it only gives suggestions to the controller for their rendering.

4.4.5 GUI Navigation

The concept of GUI navigation refers to the means by which the user moves between collections of GUI elements (between panels, groups or individual non-organizational elements), selects an on-screen control for manipulation, and the way in which this selection is represented to the user. The selection of a non-organizational element is called "setting the focus", where the focus indicates which element will receive the results of user input, such as pressing a button or entering text.

The navigation between GUI elements is handled locally by the Controller. The target device may suggest navigation rules between certain GUI elements, if this navigation order is important to the user experience. Because such navigation rules are just suggestions, the controller may need to tailor the navigation between GUI elements based on its adjustment of GUI layout.

Because the display characteristics of controllers can vary dramatically, the display of focus information is also handled locally by the controller. This might be the highlighting of a text label, a simple border around the target GUI element, or a more sophisticated graphical rendering.

4.4.5.1 Controller-Driven Navigation

If a controller is unable to render a GUI element (a panel or group element and its contents elements, or a single non-organizational element), then the controller must provide some means consistent with the capabilities of the user input and display output devices to allow the user to bring non-rendered elements into view. This process is called *controller-driven navigation.*

GUI elements may contain attributes that the controller may or may not use to guide the user during controller-driven navigation. To allow the user to control this process the controller may render items of its own choosing (e.g., arrows, scroll bars, etc.). These items are not obtained from the target and are specific to the controller implementation. However the controller implements controller-driven navigation, it must do so locally without explicit target involvement. That is, the controller may not during this process send USER ACTION messages to the target, though it (the controller) may obtain additional GUI elements from the target.

4.4.5.2 User-Driven Navigation

It is possible for GUI element data to contain non-organizational panel link elements. A panel link element has as the value of a mandatory attribute the ID of a panel element. Other attributes suggest a position and

appearance for the rendering of the panel link element. This type of element offers a means for the controller to switch from one panel to another. If the user selects this element the controller may abandon rendering the currently rendered panel, make the specified panel element the current panel, and render it appropriately. This process is called *user-driven* navigation. Again, as for controller-driven navigation, the target will only be consulted if the controller needs additional GUI elements; USER ACTION messages will not be sent by the controller.

Panel link elements thus make it possible for a target to specify GUI hierarchies of effectively arbitrary depth below the root panel.

4.4.6 Notification Scope for Target GUI Changes

As mentioned above in section 4.4.2, at any time while a controller is working with a target, the target may indicate that its GUI description has changed by sending the changed GUI element(s) to the controller via asynchronous connection. This mechanism will provide the controller with information about which parts of the target's GUI description have changed. The target may also indicate that its GUI description has changed by a USER ACTION command using this mechanism.

It is possible for the controller to request notification for only a portion of the target's GUI description, the current *notification scope*. The target will only notify the controller of changes to those target GUI elements that are within the notification scope. Target GUI elements outside of the current notification scope may change but the target will *not* send a corresponding indication to the controller. This indication is called a *change report (GUI updates)* and may refer to zero or more changed GUI elements. A change report is included in the data that is transmitted on the asynchronous connection.

Any non-organizational target GUI element that changes and is within the current notification scope will be included in the change report sent by the target to the controller. If a GUI element is added to or removed from an organizational target GUI element then that organizational element will be included in the change report sent by the target to the controller.

For low notification-traffic situations, the controller can choose to set the notification scope to be the complete target GUI description. In higher traffic situations, the controller can set the notification scope to be only the current panel. Details are in the description of the GUI UPDATE command.

The controller might discard the whole change report or a part of change report if it is not useful, e.g. when the controller has ONLY text display capability, all image data in change report are discarded.

The controller may not modify its GUI on the display using change report immediately. The controller might discard the change report except identifier of the changed GUI element when the controller is busy or doesn't have enough memory space at that time. After the controller can afford dealing with change report, the controller can get the changed GUI element using PUSH GUI DATA command.

The controller has full ownership for dealing with change report, but the panel subunit shall send the whole change report.

4.5 Indirect mode

Indirect mode simulates remote commander.

In this mode, controllers may get UI data from target device using a way other than specified in this specification, e.g. EIA-775 detailed in reference [R11] or UI data mixed with stream data, and it depends on the adequate specifications how to get and render the UI information.

The user operates a button on a physical or logical remote commander of the controller, and then the controller sends PASS THROUGH commands back to the panel subunit. In response to these commands, the panel subunit performs some action(s) if necessary. In this case, it may affect stream data e.g. mixed OSD data with stream data might be changed or the content of stream data might be changed.

Even though there may be several panels, or menus, in a target device, the device has only one panel subunit. Using target dependent manner and depending on UI data transmitting mechanism, the UI data might be switched to the next/previous panel.

In this indirect mode the panel subunit may have no source plug.

4.5.1 GUI Layout and Presentation for indirect mode

In the indirect mode, panel subunit doesn't care about the data to be displayed, and its major task is to convey user operations to a target by PASS THROUGH command.

4.5.2 Interactions for indirect mode

In the indirect mode, a controller conveys a user operation to a target by PASS THROUGH command. PASS THROUGH command can be transmitted to the target at any time. However it is target dependent whether or not the target works according to the specified operation, e.g. target device might not react to it except sending back NOT IMPLEMENTED or ACCEPTED response.

The controller may choose a target device and establish a connection between the controller display and the target device but this procedure is not specified in this specification. A connection between the controller and the target device can be any of an isochronous connection, an asynchronous connection or an analog connection.

Afterwards, the controller transfers user operations to the target by PASS THROUGH command. The target may or may not update UI data, which is transferred to the controller display by using the connection.

The interaction between a user and a target is performed by repeating the above procedure. It may be terminated when the controller disconnects the connection by user operation or other triggers.

For more detail of the PASS THROUGH command, refer to section 9.

5. Controller and Panel subunit

This clause describes the usage of panel subunit in the following three cases:

1) Peer to Peer connection; One controller with one panel subunit;
2) One controller with multiple panel subunits; and
3) Multiple controllers with one panel subunit.

5.1 Peer to peer connection

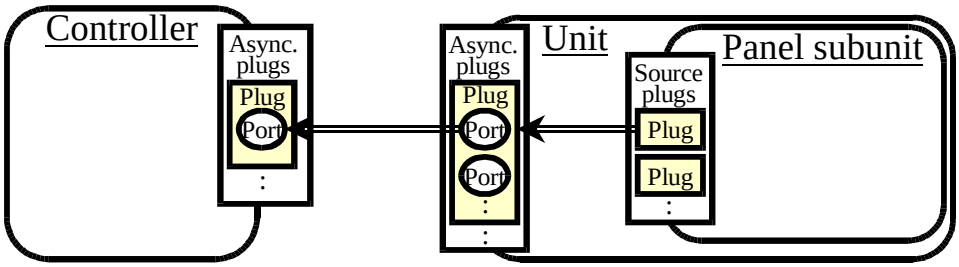

Figure 5.1 – Peer to peer connection

(1) Direct mode

In case the controller that has both AC-CTR and PS-CTR is connected to the panel subunit, the PS-CTR shall check the availability of source plugs on the panel subunit when the PS-CTR starts working with the panel subunit. If the source plug is available, the PS-CTR should issue the GUI UPDATE command with "start" subfunction and shall establish an internal connection between the asynchronous plug of the unit and the source plug of the panel subunit using CONNECT command with any available asynchronous plug option. After that, the AC-CTR shall establish an asynchronous connection between the controller unit and the target unit that contains the panel subunit in a way specified in reference [R3]. So the controller can establish the connections even if the internal connection is permanent.

After all the connections are successfully established, the controller can issue the PUSH GUI DATA command to request specific GUI information by using the source plug number of the panel subunit if needed. The panel subunit will transfer the required GUI data to the controller over the Asynchronous Connection according to the PS-CTR's requests.

The target can send GUI update information spontaneously to the controller after the PS-CTR issues the GUI UPDATE command with "start" subfunction, and it will stop right after the PS-CTR issues the GUI UPDATE command with "stop" subfunction.

If there is any user operation on GUI, the controller will decide whether or not to send user action information back to the panel subunit via the USER ACTION command.

During the period between the PS-CTR issuing "open" and "close" subfunctions, the occupied source plug of the panel subunit shall not accept any control commands from any other PS-CTR.

(2) Indirect mode

The PS-CTR doesn't care about the GUI data transfer, but the controller contained PS-CTR may make the isochronous, analog, or EIA-775 connection between the target and the controller.

If there is any user operation on a button of the remote commander, the controller will send user operation information to the panel subunit via the PASS THROUGH command.

5.2 Multiple targets

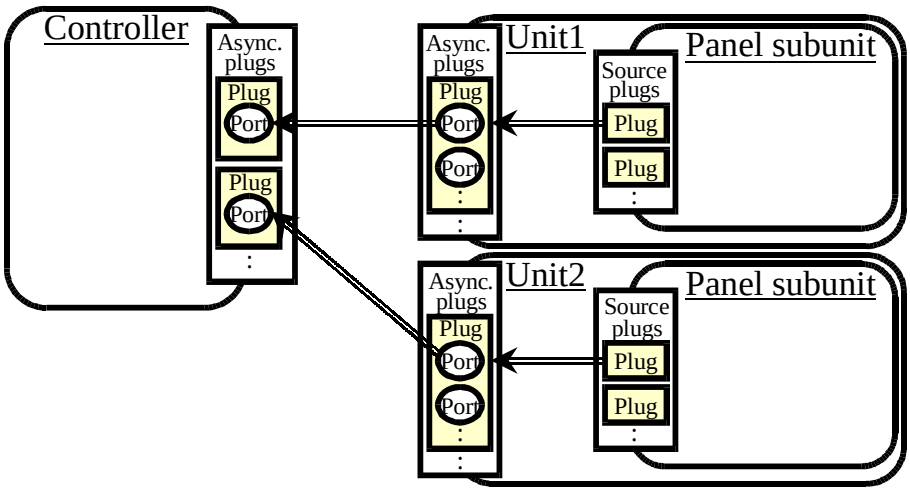

Figure 5.2 – Controller supports multiple targets

(1) Direct mode

The peer-to-peer control fashion can be extended to handle multiple targets with each target having one panel subunit inside. In this case, the controller is required to have multiple asynchronous plugs.

The controller is required to establish an internal connection between each asynchronous plug and each source plug separately using the same way as peer-to-peer connection. After that the controller is required to establish an asynchronous connection between the controller and each target separately.

The controller distinguishes the data of each panel subunit according to the asynchronous plug that receives the data.

(2) Indirect mode

The peer-to-peer control fashion can be extended to handle multiple targets with each target having one panel subunit inside. The controller may make the isochronous, analog, or EIA-775 connection between the controller and each target. It is controller implementation dependent or depend on the kind of connection how the controller distinguishes the data from which target.

5.3 Multiple controllers

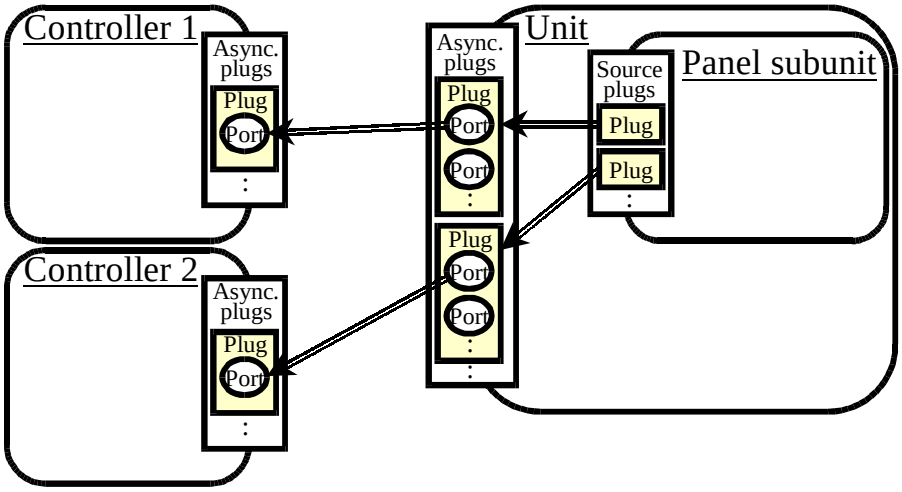

Figure 5.3 – Target supports plural controller

(1) Direct mode

The peer-to-peer control fashion can also be extended to handle multiple controllers controlling one target. In this case, the target is required to have multiple asynchronous plugs, while the panel subunit is required to have multiple source plugs.

Each controller is required to establish the corresponding internal connection between the panel subunit and one of the target's asynchronous plugs in the same way as peer-to-peer connection. Each controller is also required to establish an asynchronous connection between the target and its corresponding controller.

The panel subunit should keep consistency among the UI data for plural controllers. For instance, when the GUI element in a panel has been updated, the panel subunit should send the update to all controllers that are interested in the panel, i.e. its notification scope contains the panel.

(2) Indirect mode

The peer-to-peer control fashion can be extended to handle multiple controllers controlling one target. Each controller may make the isochronous, analog, or EIA-775 connection between the target and itself. It is target implementation dependent or depend on the kind of connection to synchronize the how the target synchronize the data to each controller.

6. Panel Subunit Data structure and transmission

This section describes the panel data structure and the panel data transmission format.

The panel data structure is used for data transfer over the asynchronous connection. The actual data structure implemented inside controller and target is vender dependent.

The data transmission format in this section is defined for panel subunit. The Panel subunit transfers the data by using this data transfer format via the asynchronous connection, which is fully explained in reference [R3].

The panel subunit does not need to support the descriptor-related commands, i.e. OPEN DESCRIPTOR COMMAND, READ DESCRIPTOR COMMAND, and WRITE DESCRIPTOR COMMAND [R1].

6.1 Panel data structure

The panel subunit model defines two organizational GUI elements, i.e. PANEL and GROUP, and a GROUP is optional. Any of PANELs and GROUPs can be modifiable, if the panel subunit is implemented to support this feature. In some cases it makes sense to allow modifications, such as changing the Icon or Button in the panel; however, this is completely up to the panel subunit implementation.

The panel subunit model defines some non-organizational GUI elements, i.e. BUTTON, ICON or so on, called element. Any of elements can be modifiable, if the panel subunit is implemented to support this feature. In some cases it makes sense to allow modifications, such as changing the bitmap or sound data; however, this is completely up to the panel subunit implementation. Each element is linked from an organizational GUI element, and the element can be linked from plural organizational GUI elements, i.e. panel 1 and panel 2 can contain the same element (ex. icon).

The panel subunit model defines some data entities, i.e. Text, Bitmap and Sound. Each data entity is linked from an organizational GUI element or a non-organizational GUI element. One data entity may be linked from multiple organizational GUI elements and/or non-organizational GUI elements.

The following diagram illustrates the general relationship among the panel subunit elements.

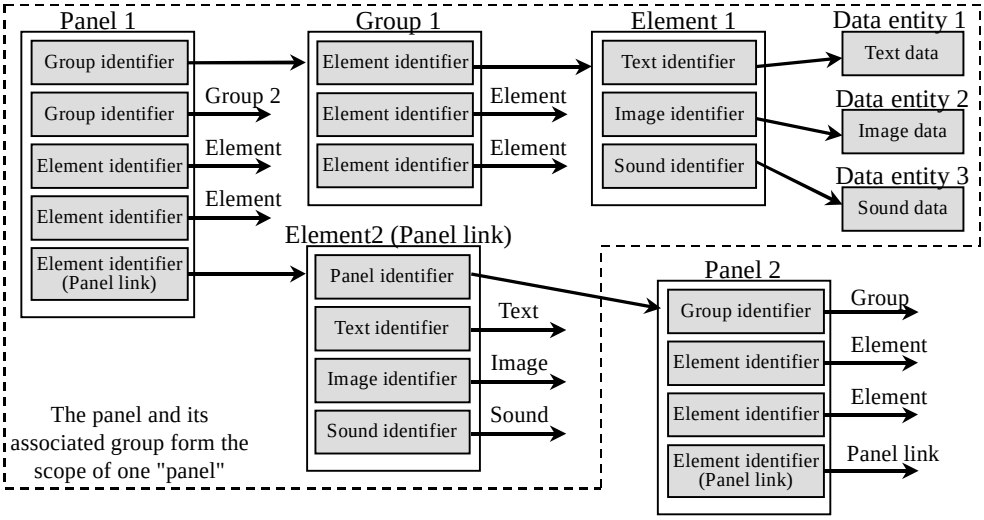

Figure 6.1 – Panel data structure

As shown in Figure 6.1, panel 1, which is composed of three free-standing elements and two groups, is the root panel which forms the top of the panel hierarchy. Element 2 (a Panel link) is a link to refer to another panel: panel 2. Group 1, which is part of panel 1, is composed of three individual GUI elements.

An important part of this model is that a panel and its directly associated groups (as shown bounded by the dashed-line box in the diagram) form the scope of a “panel”. This scope is important for changing and specifying the scope of notification, i.e. GUI updates.

The user operation for each GUI element is transmitted by the USER ACTION command, and the PASS THROUGH command could be used for a short cut command or for cursor movement inside an icon.

The panel data could consist of only one icon, which has the bitmap to be displayed on the controller screen.

In this case, the user operation could be transmitted by the PASS THROUGH command.

6.2 GUI data Transmission

The Panel subunit transfers GUI data on asynchronous connection. The transmission format is defined as a generic but simple data structure with extensibility in the panel direct mode specified in this panel subunit specification.

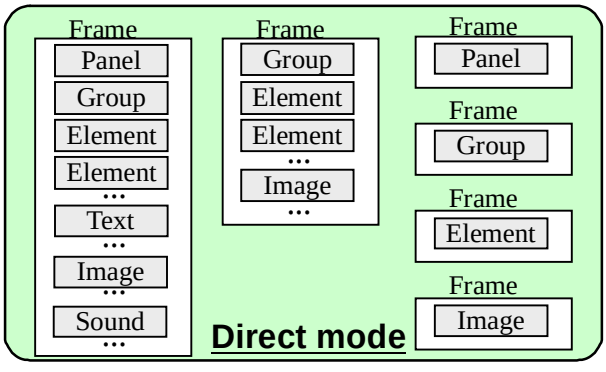

Figure 6.2 – Data in a frame of asynchronous connection

As shown in the above figure, the panel subunit can transfer the GUI data in different scopes inside one frame specified in asynchronous connection, e.g. one frame can contain arbitrary set of GUI elements.

In case of the change notification, if more than one GUI element or data entity has changed at the same time, then the panel subunit shall send those GUI updates in the same asynchronous connection frame.

For example, if the PANEL GUI data has been changed, the asynchronous connection frame, which the panel subunit sends to the controller, should contain the update information in the following order:

1) the changed PANEL;
2) the changed element(s) following the order of the element list in the PANEL;
3) the changed element(s) in the GROUP, should follow the order of the element list in the GROUP, and each GROUP is chosen by the order of the element list in the PANEL ;
4) the changed text content(s), which is put in the reverse order of elements and Panel, that is, the order of the changed text content(s) begins with the changed text content associated to the last changed element in (2);
5) the changed bitmap content(s), which is put in a similar order of the changed text contents;
6) the changed audio content(s), which is put in a similar order of the changed text contents.

If the specified element/content above is not changed, the clause should be skipped.

This order should be used to specify the GUI update data for GROUP or each GUI element, and it should also apply to the response to the PUSH GUI DATA command as well.

Note that

– In case the element(s) has been deleted or replaced, the parent element should be sent first in the asynchronous connection frame because the element list in the parent element is changed. For example, when the element in the PANEL is replaced by another element, the asynchronous connection frame should have the PANEL first, followed by the replaced new element and its content(s).
– In case the attribute(s), NOT the link information (bitmap link, etc.), in the element has been changed, only the element should be sent. For example, when the attribute value of an element in the PANEL is changed, only the element should be sent because no data entity is changed.

6.3 Notification Scope

The controller can specify a scope for GUI updates, which is called notification scope.

By using GUI UPDATE control command, it is possible for the controller to request notification, which contains the changed GUI elements, only for the *notification scope*: current panel or device

When the target changes some GUI elements, then the one or more changed GUI elements are spontaneously transferred to the controller in one frame using asynchronous connection. GUI elements outside of the current notification scope may change but the target will *not* send a corresponding indication to the controller.

In case the notification scope is current panel, the panel subunit will notify the controller of the changed GUI elements contained only within the current panel. The current panel is changed when the identifier of the other panel is specified in PUSH GUI DATA control command.

In case the notification scope is device, the panel subunit will notify the controller of the changed GUI elements in all of the target's GUI descriptions (i.e. all panels in the device).

6.4 Generation number

The generation number is defined in panel subunit and shows the GUI generation in the target. The value of the generation number is made by a sixteen bits counter and incremented when certain state/status related to the GUI inside the device has changed. The generation number is used for the notification of GUI element's changes inside the device and in commands from the controller. This is useful to check whether or not the GUI related information in the target has changed.

The generation number shows changes in the target. That is, whenever any GUI element or data entity in a panel, has been changed, the device's generation number is incremented by one, the generation number of that panel is set to the device's generation number, and the generation number of any other panel shall NOT change.

When the target changes some GUI elements, depending on the notification scope, one or more changed GUI elements with suitable generation number are spontaneously transferred to the controller in a frame of the asynchronous connection.

In case the notification scope is the current panel, the panel subunit will notify the controller of the changed GUI elements only inside the current panel with the current panel's generation number.

In case the notification scope is device, the panel subunit will notify the controller of all the changed GUI elements in the target's GUI description with device's generation number.

The controller can check whether the panel information which stored in the controller is last generation or not by using the generation number. It is important whether a generation number stored in the controller is the same as one responded from the target. But the numeric number itself is not so much important.

When the user operates a GUI element, the controller sends USER ACTION control command with the last generation number to the panel subunit. So the panel subunit can find whether the panel that the user interact with is the last one or not. It is target dependent whether the target accepts the user action or not, if the panel is not the last one, for example if the previous change is not related to the GUI element that the user interacts with, the target accepts the user action for this GUI element.

Table 6.1 – Example of generation number

	Device's generation number	Panel A's generation number	Panel B's generation number
Initial	1	1	1
Element in Panel A was changed	2	2	1
Element in Panel A was changed	3	3	1
Element in Panel B was changed	4	3	4
Element in Panel A was changed	5	5	4
:	:	:	:

Table 6.1 shows an example of generation number change. If the controller subscribes changes of panel A as its current panel, and a GUI element in panel A is changed, then the device's generation number is

incremented and panel A's generation number is set to the new device's generation number. The changed GUI element with new panel A's generation number is sent to the controller on asynchronous connection.

If the controller subscribes changes of panel A, and the GUI element in panel B is changed, then the device's generation number is incremented, but the panel subunit shall not notify this change to the controller.

6.4.1 Generation number value

The initial value of generation number should be set to one. When the generation number is incremented up from one to $FFFE_{16}$, the panel subunit shall set all generation number values in the target to one and shall send the generation number to each controller.

When the generation number is zero, the panel subunit shall never change any GUI elements within the panel.

When the generation number is $FFFF_{16}$, the information with this generation number doesn't affect GUI elements and their structure but with some new value, e.g. for show range. This information is called status report. For details, refer to section 7.9.

The generation number $FFFF_{16}$ is also used for the suggestion of the next panel to be displayed. This information is called the suggested data. For details, refer to section 7.8

7. Basic Panel Data format

This section describes the basic data format used by the panel subunit. The panel data format is independent of the data format implemented inside the target or the controller.

7.1 Identifier format

Each GUI element has an identifier called element identifier.

byte count	contents
00 00_{16}	element_type
00 01_{16}	
00 02_{16}	element_ID
00 03_{16}	

Figure 7.1 – Element identifier description

The *element_type* field describes the meaning and format of the GUI element and data entity, and the values of *element_type* are specified in Table 8.1, Table 8.2, and Table 8.3.

The *element_ID* field contains the sixteen bits numeric number that uniquely identifies the GUI element and data entity within the panel subunit except value of $FFFF_{16}$.

7.2 Data transfer format on asynchronous connection

The transfer format is defined as a generic but simple data structure with extensibility. The panel subunit transfers the data using this transfer format in a frame of the asynchronous connection.

The following figure defines the frame format for panel subunit:

byte count	contents
00 00_{16}	type_code
00 01_{16}	total_length
00 02_{16}	
00 03_{16}	
00 04_{16}	frame_dependent_info (generation_number)
00 05_{16}	
00 06_{16}	panel_specific_data (element data block(s) or none)
:	
:	

Figure 7.2 – Asynchronous connection frame format – panel basic structure

The *type_code* field specifies the types of the asynchronous connection frame.

The following table contains the definitions of the type_code field value.

Table 7.1 – Type code for asynchronous connection frame

type code	type_code value	comment
	00_{16}-$7F_{16}$	not used in this specification
pushed data	80_{16}	The data according to the request of PUSH GUI DATA command
updated data	81_{16}	The updated data according to the request of notification
suggested data	82_{16}	The data that the target suggests the controller to display on the screen
status report	83_{16}	The status report for GUI element
83_{16} – FF_{16}	84_{16} – FF_{16}	Reserved for future extension

The *total_length* field describes the number of bytes for the remainder of this frame, which is all fields following this *total_length* field. When the *total_length* field has a value of zero, there is no information or data in the frame.

The *frame_dependent_info* field contains information that is specific to each frame data. The *frame_dependent_info* field contains only a generation number, which describes the generation related to the contents of the following panel specific data.

The *panel_specific_data* field contains arbitrary set of element data blocks and each element data block contains either a GUI element or a data entity. When the panel subunit sends only generation number to the controller, e.g. it sets the generation number to one, it shall not send the *panel_specific_data* field, i.e. the frame has only *type_code, total_length* field and *frame_dependent_info* field which contains a generation number. In this case the *type_code* field value is set to 81_{16} (updated data).

In case of "suggested data" and "status report", refer to the section 7.8 and 7.9.

7.3 Element data block data format

Each GUI element or data entity has the format described in Figure 7.3.

byte count	contents
00 00_{16}	element_identifier
00 01_{16}	
00 02_{16}	
00 03_{16}	
00 04_{16}	element_data (GUI element or data entity)
:	
:	
:	

Figure 7.3 – Element data block – panel basic structure

The *element_identifier* field contains an element identifier, which uniquely identifies the GUI element or data entity.

The *element_data* field contains either GUI element using a GUI element format or data entity using data entity format.

7.4 GUI element basic format

Panel subunit supports both organizational and non-organizational GUI element: PANEL, GROUP, BUTTON, ICON and so on. Each GUI element is described by using the GUI element format structure. The GUI element will be put in the *element_data* area within the element data block.

All GUI elements will have the same basic layout, which includes some standard fields at the beginning, followed by a collection of attributes and references to data entities and/or the other GUI elements.

byte count	contents
$00\ 00_{16}$	element_length
$00\ 01_{16}$	
$00\ 02_{16}$	size_of_mandatory_attributes
$00\ 03_{16}$	list_of_mandatory_attribute
:	
:	
:	number_of_mandatory_links (n)
:	mandatory_link[0]
:	
:	
:	
:	:
:	mandatory_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 7.4 – GUI element basic format

The *element_length* field contains the number of bytes which follow in this GUI element basic structure. The value of this field does *not* include the *element_length* field itself.

The *size_of_mandatory_attributes* field specifies the number of bytes used for the following *list_of_mandatory_attribute* field. The size field is one byte, and is NOT included in this calculation.

If the *list_of_mandatory_attribute* field doesn't exist, the *size_of_mandatory_attributes* field shall be set to zero.

The *list_of_mandatory_attribute* field contains information that is specific to a particular *element_type* in *element_identifier*, and this field value is defined in each GUI element. Refer to each GUI element definition in section 10.5 for details on this field.

The *number_of_mandatory_links* field specifies the number of mandatory links used for the following *mandatory_link[x]* field. If the *mandatory_link[x]* field doesn't exist, the *number_of_mandatory_links* field shall be set to zero.

The *mandatory_link[x]* field is a mandatory link to the other element, i.e. GUI element or data entity, and this field value is defined in each GUI element except for panel and group GUI element. Refer to each GUI element definition in section 10.5 for details on this field.

The *size_of_optional_attributes* field specifies the number of bytes used for the following *list_of_optional_attribute* field. The size field is one byte, and is NOT included in this calculation.

If the *list_of_optional_attribute* field doesn't exist, the *size_of_optional_attributes* field shall be set to zero.

The *list_of_optional_attribute* field contains information that is specific to a particular *element_type* in *element_identifier*, and this field value is defined in each GUI element, and the possible value of this field is defined in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field specifies the number of optional links used for the following *optional_link[x]* field. If the *optional_link[x]* field doesn't exist, the *number_of_optional_links* field shall be set to zero.

The *optional_link[x]* field is a optional link to the other element, i.e. GUI element or data entity, and the possible value of this field is defined in Table 10.2 – Optional info_type field value.

7.5 Data entity format

Panel subunit supports the data entity: text, image and sound. Each data entity is described by using the data entity format structure. The data entity will be put in the *element_data* area within the element data block.

byte count	contents
$00\ 00_{16}$	data_length
$00\ 01_{16}$	
$00\ 02_{16}$	
$00\ 03_{16}$	
$00\ 04_{16}$	data
:	
:	

Figure 7.5 – Data entity format

The *data_length* field specifies the number of bytes of the data field, and this field value is the same as the data_entity_size attribute value if it exists in GUI element.

The *data* field contains text data, image data or sound data. A text data is a string, encoded as Unicode specified in reference [R8], an image is a bitmap file format, encoded as PNG specified in reference [R9] for instance, and a sound is an audio file format, encoded as AIFF-C specified in reference [R10] for instance. Furthermore, an image can be a YCbCr bitmap format specified in section 8.2.3.2.

7.6 Pushed data

The pushed data is used to send the data according to the request of PUSH GUI DATA command. The data that requested by one PUSH GUI DATA command shall be transferred by one frame of the asynchronous connection.

7.7 Updated data

The updated data is used to send the updated data and it is spontaneously transferred from the target. The data that changed in the target at the same time should be transferred by one frame of the asynchronous connection. The updated data shall be in the notification scope except HELP PANEL, ALERT PANEL, DEVICE ICON and VENDOR ICON.

Note that the updated data may contain the PANEL but it doesn't mean the change of the notification scope. The notification scope may be changed only when the controller pulls a PANEL.

7.8 Suggested data

The Suggested data is used for the suggestion from the target. The target can suggest the panel that the target wants for the controller to display on the screen in order for the user to notice something, e.g. alert panel or so on.

Note that the controller finally decides which panel should be displayed on the screen. After the decision, the controller issues the PUSH GUI DATA control command to get the panel data and displays it.

The transfer format of suggested data is defined as below:

byte count	contents
00 00_{16}	type_code (82_{16})
00 01_{16}	total_length
00 02_{16}	
00 03_{16}	
00 04_{16}	frame_dependent_info (generation_number: $FFFF_{16}$)
00 05_{16}	
00 06_{16}	suggeted_new_panel
00 07_{16}	
00 08_{16}	
00 09_{16}	
:	delete_panel[0]
:	
:	
:	
:	:
:	delete_panel[n-1]
:	
:	
:	

Figure 7.6 – Asynchronous Connection Frame Format – suggested data

The *type_code* field specifies the type of the frame, and the *type_code* field shall be set to 82_{16} in the frame in case of suggested data.

The *total_length* field describes the number of bytes for the remainder of this frame, which is all fields following this *total_length* field.

The *frame_dependent_info* field contains only a generation number that shall be set to $FFFF_{16}$ in case of suggested data.

The *suggeted_new_panel* field contains the element identifier, defined in section 7.1, of the panel, which the panel subunit suggests displaying on the screen. In case *suggeted_new_panel* field value is $FFFF_{16}$, it suggests finishing GUI representation, i.e. the controller doesn't break connection to the panel subunit but disappear GUI from its screen.

The *delete_panel[x]* field is an element identifier, defined in section 7.1, of the panel, which is no longer available at the panel subunit. This field may not exist in the frame, but the *suggeted_new_panel* field is requested if any *delete_panel[x]* field exists.

7.9 Status report

The status report is used for sending the new status of the GUI element to the controller, e.g. the current value for show range like level meter.

The transfer format of status report is defined as below:

byte count	contents
00 00_{16}	type_code (83_{16})
00 01_{16}	total_length
00 02_{16}	
00 03_{16}	
00 04_{16}	frame_dependent_info (generation_number: $FFFF_{16}$)
00 05_{16}	
00 06_{16}	report_data
:	
:	

Figure 7.7 – Asynchronous Connection Frame Format – status report

The *type_code* field specifies the type of the frame, and the *type_code* field shall be set to 83_{16} in case of status report.

The *total_length* field describes the number of bytes for the remainder of this frame, which is all fields following this *total_length* field.

The *frame_dependent_info* field contains only a generation number that shall be set to $FFFF_{16}$ in case of status report.

The *report_data* field contains new status value of the attribute of GUI element.

The *report_data* field is defined below:

byte count	contents
$00\ 00_{16}$	element_identifier
$00\ 01_{16}$	
$00\ 02_{16}$	
$00\ 03_{16}$	
$00\ 04_{16}$	new_status
:	

Figure 7.8 – The report data – new status data

The *new_status* field contains the new status value of the specified GUI element in *element_identifier* field using an attribute structure defined in section 10.4.1.

In report data, *element_identifier* shall point to one of SHOW RANGE, SET RANGE, STATUS and TOGGLE BUTTON. The *new_status* field shall be *value_set* field specified in SHOW RANGE or SET RANGE, *status_woking* field specified in STATUS, *toggle_status* field specified in TOGGLE, *content_availability* field specified in CONTENT ICON.

8. GUI elements and data entity

This section describes the kind of the GUI elements and data entities.

8.1 GUI element

Each organizational and non-organizational GUI element is identified by using element identifier that has element_type and element_ID.

Each element_type field structure for GUI element is defined below:

byte count	msb							lsb
00_{16}	reserved	0	reserved					
01_{16}	GUI_element_type							

Figure 8.1 – Element_type for GUI element

The *GUI_element_type* field describes the meaning and format of the GUI element, and the value of *element_type* is specified in Table 8.1, Table 8.2.

8.1.1 PANEL and GROUP

The following table contains the definitions of the element type value for organizational GUI element.

Table 8.1 – GUI element type value for PANELs and GROUP

organizational GUI element	GUI_element_type value	comment
PANEL	00_{16}	The panel can contain the groups and free-standing GUI elements.
HELP PANEL	01_{16}	The help panel has the same structure as a panel.
ALERT PANEL	02_{16}	The alert panel has the same structure as a panel.
GROUP	03_{16}	The group can contain free-standing GUI elements.
The other	$04_{16} – 0F_{16}$	Reserved for future extension
	$10_{16} – FF_{16}$	Specified in Table 8-2

PANEL, HELP PANEL, ALERT PANEL and GROUP are non-interactive GUI element; no user action is defined for them.

8.1.1.1 PANEL

There can be any number of these PANELs in a panel subunit, but only ONE can be a root panel; all others are not directly linked from root panel but referred by PANEL LINK element in the root panel.

A PANEL can contain GROUP(s) or non-organizational GUI element(s) in link fields.

Each GROUP and each GUI element are referred by the corresponding element identifier, which consists of element_type and element_ID.

A PANEL can also have data entities directly, e.g. link to a background picture.

A PANEL has some mandatory attributes and can also have optional attributes. For details, refer to section 10.5.1.

8.1.1.2 HELP PANEL

A HELP PANEL is used to show the help information to the user and it has the same structure as the PANEL. The controller pulls the HELP PANEL when it decides to get help information of the GUI element using its help panel ID attribute.

Note that HELP PANEL is always out of the notification scope, i.e. the current panel shall not be changed even though the controller pulls HELP PANEL.

8.1.1.3 ALERT PANEL

An ALERT PANEL is used to show the alert information to the user and it has the same structure as the PANEL. The target sends the ALERT PANEL when it decides to display alert information to the user.

Note that ALERT PANEL is always out of the notification scope, i.e. the current panel shall not be changed even though the controller pulls ALERT PANEL.

8.1.1.4 GROUP

There can be any number of these GROUPs in a PANEL, and these are directly linked from the PANEL.

A GROUP can contain one or more non-organizational GUI elements in link fields.

Each GUI element is referred by the corresponding element identifier, which consists of element_type and element_ID.

The element identifier (element_type and element_ID) of a GROUP in a PANEL points down to the GROUP.

There can be any number of GROUPs in a PANEL. GROUPs are always below a PANEL in the hierarchy. As GROUP is optional, a PANEL may have no GROUP at all.

A GROUP can also have data entities directly, e.g. link to a background picture.

A GROUP has some mandatory attributes and can have optional attributes, For details, refer to section 10.5.2.

In case the controller doesn't display one or more GROUPs but PANEL, the information which describe GROUP itself, e.g. background color, background picture, background pattern, audio video, title and group label, should not be displayed on the screen.

8.1.2 Non organizational GUI elements

The following table contains the element_type definitions of non-organizational GUI elements.

Table 8.2 – GUI Element type value for non-organizational GUI elements

non-organizational GUI element	GUI_element_type value	user action (if the GUI element is interactive)	comments
PANEL LINK	10_{16}	select	Image which shows the linked PANEL
BUTTON	11_{16}	press and release	Press/Release button
ANIMATION	12_{16}	select_element	Pseudo animation
SHOW RANGE	13_{16}	select	Level meter, indicator, etc.
SET RANGE	14_{16}	set_value	Slider, Dial, etc.
ENTRY	15_{16}	enter_data	Text or numeric entry
CHOICE	16_{16}	choose_list	Choose one or plural element
TEXT	17_{16}	select	Text data for display
STATUS	18_{16}	no user action	Object to show busy status
ICON	19_{16}	select	Image for selection or display only
TOGGLE	$1A_{16}$	select_item	n-states toggle button
Reserved	$1B_{16}$-$3F_{16}$		Reserved for future definition
CONTENT ICON	40_{16}	select	Image which show the contents
DEVICE ICON	41_{16}	no user action	Image which show device
VENDOR ICON	42_{16}	no user action	Image which show vendor
Reserved	43_{16}-FF_{16}		Reserved for future definition

Most of all GUI elements support *interactive* attribute. If the GUI element is not interactive specified in *interactive* attribute, then the GUI element is not interactive and the controller shall not sent any user action to the target. Note that the *interactive* attribute value may be changed dynamically.

8.1.2.1 GUI Element specific information

All GUI elements have the GUI element format structure; attributes and links of the GUI element are in GUI element format structure. The attributes and the links are detailed in each GUI element at section 10.2.

The controller can get a DEVICE ICON, CONTENT ICON or VENDOR ICON from each panel subunit without PANEL or GROUP information using PUSH GUI DATA command and these icons could also be in the panel data structure. The other non-organizational GUI elements are contained in the panel data structure.

8.2 Data entity

Each data entity is identified by using element identifier that has element_type and element_ID.

Each element_type field structure for data entity is defined below:

byte count	msb							lsb
00_{16}	reserved	1	reserved		data_format			
01_{16}	info_type							

Figure 8.2 – element_type structure for data entity

The *data_format* field describes the format of the data entity, and the value of *data_format* is specified in Table 8.3.

The *info_type* bit describes optional property for data entity, refer to section 10.2.

8.2.1 Data formats

Panel subunit supports the data entity: text, image and sound. The following table contains the data type and format definitions for panel subunit-specific element:

Table 8.3 – Data format value for data entity

Data entity	Data_ format value	Data format type	comments
Text	0000_2	UNICODE	Detailed in reference [R8]
	0001_2 – 0011_2	all other text data format	Reserved for future definition
Image	0100_2	PNG bitmap data	Limited to color type =3 and bit length =1,2,4 or 8 in reference [R9]
	0101_2	HAVi raw bitmap	Detailed in reference [R12]
	0110_2	YCbCr bitmap data	Detailed in section 8.2.3.2
	0111_2	all other Image data format	Reserved for future definition
Sound	1000_2	AIFF-C	Limited to sample size = 8 Sample rate = 22.050kHz in reference [R10]
	1000_2 – 1011_2	all other sound data format	Reserved for future definition
Reserved	1100_2 – 1111_2	all other data format	Reserved for future definition

Data entities are non-interactive; no user action is defined for them.

8.2.2 Text data

The text data has the data entity format structure and used for the textual description of the GUI element or just information to the user on the display of the controller.

All text data in the panel subunit will be in UNICODE [R8] UTF-16 and ISO/IEC 10646-1 [R15] UCS-2.

8.2.3 Image data

The image data has the data entity format structure and used for the graphical description of the GUI element or just information to the user on the display of the controller.

PNG, HAVi raw bitmap and YCbCr data is used for panel subunit bitmap format in the *data* field of the data entity, however panel subunit doesn't support full specification of PNG or YCbCr;

8.2.3.1 PNG bitmap data

The detail is specified in reference [R9]. The limitation for the PNG bitmap data is specified below.

Limitations on PNG: color type = 3 (only index color is permitted)

bit depth = 1,2,4,8 (max. 8bit color)

8.2.3.2 YCbCr bitmap data

The *data* field in data entry format specified in section 7.5 can have a YCbCr bitmap format below.

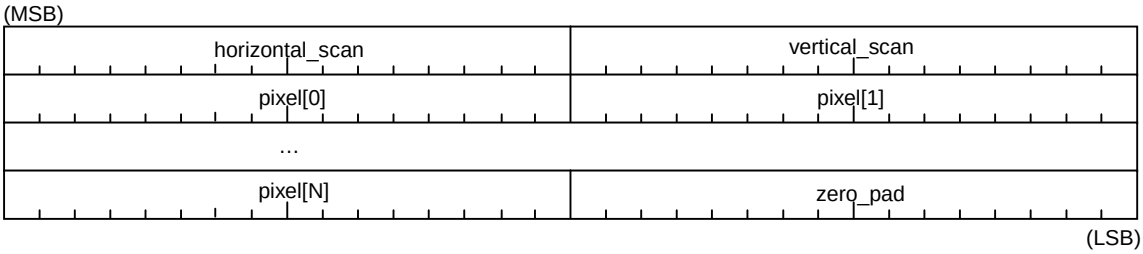

Figure 8.3 – YCbCr bitmap format

The horizontal_scan field specifies the horizontal number of pixels in each of bitmap.

The vertical_scan field specifies the vertical number of pixels in each of bitmap.

The pixel[0] through pixel[N] are sixteen bit pixel values using a packed YCbCr format specified below. The pixels shall be listed in a scan order that is left to right and top to bottom, and the left to right order is prior to the top to bottom order.

Figure 8.4 – Packed YCbCr format

The *alpha* field value is defined as follows; Value zero indicates that the pixel is entirely transparent while a value of three indicates that the pixel is opaque. The other values represent intermediate transparencies approximately as fractional opacity, alpha/3.

The *y* field value specifies the luminance of the pixel, where zero is off and $3F_{16}$ represents full intensity. The other values represent intermediate intensities approximately as a fraction of 63.

The *cb* and *cr* field value specifies the intensity of their respective chroma components, where zero is off and F_{16} represents full intensity. The other values represent intermediate intensities approximately as a fraction of 15.

8.2.4 Sound Data

The sound data has the data entity format structure and used for the click sound, alarm sounds, BGM(back ground music), etc.

AIFF-C sound data is used for panel subunit sound format, however panel subunit doesn't support full specification of AIFF-C;

Limitations on AIFF-C: sample size = 8 bits
sample rate = 22.050kHz

9. Panel Subunit Commands

The address of the recipient of AV/C commands is defined by the *subunit_type* and *subunit_ID* specified in the reference [R1]. In case of the panel subunit, the *subunit_type* is 09_{16}.

The panel subunit controller should send any command to the panel subunit only after it gets the final response of the previous command from the panel subunit except for bus reset case and "cancel" subfunction of PUSH GUI DATA command.

The following table defines the commands which are specific to the direct mode of the panel subunit:

Table 9.1 – Panel subunit commands for direct mode

Opcode	Value	Support level (by *ctype*)			Comments
		C	S	N	
GUI UPDATE	$7D_{16}$	M	M	-	Allows the controller to request GUI update when GUI element changes.
PUSH GUI DATA	$7E_{16}$	M	M	-	Allows the controller to inquire about the status of a panel, to request GUI data via asynchronous connection.
USER ACTION	$7F_{16}$	O*	–	–	Used to transfer user action information from controller to panel subunit.
PASS THROUGH	$7C_{16}$	O*	–	–	Used to transfer user operation information on a button from controller to panel subunit.

It is strongly recommended that panel subunit controller supports USER ACTION control command and PASS THROUGH control command for interoperability among devices in the direct mode. The panel subunit shall support PASS THROUGH command in case that the GUI element has the *cursor_through* attribute specified in section 10.4.3.2.20.

The following table defines the command which is specific to the indirect mode of the panel subunit:

Table 9.2 – Panel subunit command for indirect mode

Opcode	Value	Support level (by *ctype*)			Comments
		C	S	N	
PASS THROUGH	$7C_{16}$	M	–	–	Used to transfer user operation information on a button from controller to panel subunit.

If the panel subunit supports the indirect mode, the panel subunit shall not support GUI UPDATE control/status command, PUSH GUI DATA control/status command and USER ACTION control command.

9.1 GUI UPDATE control/status command

The GUI UPDATE control/status command allows a controller to request GUI update for changes in GUI elements. This command also allows a controller to check the capability, i.e. generation ID and GUI resolution, of the panel subunit. This command is defined as both a CONTROL and a STATUS command.

The format of the GUI UPDATE control/status command is as follows:

	msb							lsb
opcode	GUI UPDATE ($7D_{16}$)							
operand[0]	source_plug							
operand[1]	generation_ID							
operand[2]	GUI_resolution							
operand[3]	subfunction							
operand[4]	scope							

Figure 9.1 – GUI UPDATE command format

The *source_plug* field specifies the source plug number, and this source plug will output the GUI data. If the number of source plugs is N (N=0, 1, ..30), the panel subunit has no source plug (N=0) or source plugs: [0], [1], ... and [N-1].

The controller becomes the owner of a source plug if the controller gets the ACCTEPT response of GUI UPDATE command with “open” subfunction.

The *generation_ID* field describes which AV/C panel subunit elements are used by this subunit. This field can have one of the values in the following table.

Table 9.3 – Generation ID value

generation ID value	comments
00_{16}	GUI elements set, format and commands defined in this panel subunit specification version 1.0
all others	Reserved for future definition

The *GUI_resolution* field specifies the GUI resolution(s) supported by the panel subunit. The values of the *GUI_resolution* field are defined in the following table.

Table 9.4 – GUI resolution

GUI_resolution value	comments
xxxx $xxx1_2$	The safety display area size is 640 x 480 pixels (mandatory)
xxxx $xx1x_2$	The safety display area size is 960 x 540 pixels
all others	Reserved for future definition

In case of GUI UPDATE status command, the panel subunit shall return the capability by setting all supported GUI resolution bits. In case of GUI UPDATE control command, the panel subunit controller shall set only one bit of *GUI_resolution* value.

Any panel subunit that supports the direct mode shall support panel data based on the 640 x 480 pixels safety display area. The safety display area is the area specified for the whole GUI data to be displayed on the controller's screen and the target should make all panels within the size of the safety display area.

In case of 640 x 480 pixels, the aspect ratio of the GUI data is specified in the aspect ratio attribute defined in section 10.4.2.3. In other cases, the aspect ratio of the GUI data is intended to square pixel.

The *scope* field describes the notification scope of the panel subunit. When the controller invokes this command, the *scope* field shall be set to the requested notification scope.

The following table contains the scope definitions for panel subunit.

Table 9.5 – Scope value

scope value	scope	comments
xxxx $xxx1_2$	current panel	Panel subunit notifies all GUI element changes in the current panel.
xxxx $xx1x_2$	add	Panel subunit notifies all GUI element changes for all panels retrieved by the controller.
xxxx $x1xx_2$	device	Panel subunit notifies all GUI element changes in the target.
xxxx $1xxx_2$	content icons	Panel subunit notifies all CONTENT ICONs' changes.
all other values		Reserved for future definition

In case of GUI UPDATE status command, the panel subunit shall return the capability by setting all bits for the supported scopes. In case of GUI UPDATE control command, the panel subunit controller shall set only one bit of the *scope* value. The panel subunit that supports the direct mode shall support one or more scopes.

The current panel is the latest retrieved panel by the controller in case of "current panel", "add" and "device" scope. On the "content icons" scope, the current panel is not defined.

In case of "current panel" and "add" scope, the current panel is not identified until the panel subunit invokes the first PUSH GUI DATA control command.

The *subfunction* field describes the subfunctions of this command. The following table contains the GUI update subfunction definitions for panel subunit.

Table 9.6 – Subfunction value

subfunction value	subfunction	comments
00_{16}	open	Panel subunit opens the source plug and assigns the ownership to the controller.
01_{16}	close	Panel subunit closes the ownership for the source plug.
02_{16}	restore	Panel subunit restores the ownership for the source plug after each bus reset.
03_{16}	start	Panel subunit starts to notify the GUI changes in the indicated scope.
04_{16}	stop	Panel subunit stops GUI update notification of the GUI updates.
05_{16}	change	Panel subunit changes the notification scope of the GUI updates.
all other values		Reserved for future definition

When the controller wants to start working with the panel subunit, the “open” subfunction is used to obtain the ownership of one source plug. In case of the “open” subfunction, the *scope* field shall be set to 00_{16}. After that, the controller makes the connections between the source plug of the panel subunit and itself.

After the connections are established, the controller can issue the “start” subfunction to start GUI update notification on the asynchronous connection.

The “change” subfunction is used to change the GUI update notification scope.

The “stop” subfunction is used to stop the GUI update notification, and the controller may invoke the “start” subfunction to the panel subunit after the “stop” subfunction with the source plug number that the controller owns.

The “close” subfunction is used when the controller finishes working with the panel subunit, and the controller lost the ownership of the source plug if accepted. The “close” subfunction is valid after the “stop” subfunction.

The “restore” subfunction is used for resumption of the ownership for the source plug after bus reset. To keep consistency with RESERVE control command defined in [R1], the resumption shall be completed within ten seconds after bus rest.

The “close”, “restore”, “start”, “stop” and “change” subfunction is valid for only owner of this source plug.

The “change” subfunction is used to change the GUI update notification scope.

- When the scope is changed to “current panel” scope from the “add” or “device” scope by using “change” subfunction, the current panel is set to the latest retrieved panel.
- When the scope is changed to “add” scope from the “add” scope by using “change” subfunction, the current panel is set to the latest retrieved panel and the notification scope restarts with the current panel.
- When the scope is changed to “device” scope from the “add” or “current” scope by using “change” subfunction, the current panel is set to the latest retrieved panel.
- When the scope is changed to “content icons” scope by using “change” subfunction, the current panel is defined as the latest retrieved panel and the notification scope is “content icons”.

– When the scope is changed from the “contents icons” scope by using “change” subfunction, the “current panel” is not identified. So GUI updates shall be suspended until next PUSH GUI DATA control command with a panel’s element type value.

Table 9.7 – New notification scope for change subfunction

New scope	previous scope			
	current	add	device	content icons
current	current panel	←	current panel to be retrieved	←
add	current panel and panels to be retrieved	cancel the previous scope and restart with current panel and panels to be retrieved	panels to be retrieved	←
device	all panels	←	←	←
content icons	all content icons	←	←	←

Note: “← “ means to keep the same scope.

9.1.1 GUI UPDATE control command

GUI UPDATE command may be used as a control command to request a GUI update notification for GUI changes and notification scope of the source plug. So the controller shall specify the source plug number and scope.

The following table shows the value of each field in the CONTROL command, ACCEPTED response and REJECTED response.

Table 9.8 – Command and response frame of GUI UPDATE control command

field	CONTROL command frame	REJECTED response frame	ACCEPTED response frame
source_plug	any available source plug number (FF_{16}) (open), or specified source plug number	←	assigned source plug number (open), or specified source plug number
generation_ID	generation_ID value (00_{16})	←	←
GUI_resolution	specified GUI resolution value	←	←
scope	specified scope value	←	←
subfunction	subfunction value	←	←

Note: “← “ means the same value as the previous frame

If the panel subunit supports this command and doesn’t permit execution of the command at the time after receiving the command, it returns the REJECTED response.

If the panel subunit supports only the indirect mode, the NOT IMPLEMENTED response shall be returned anytime for the GUI UPDATE control command.

9.1.2 GUI UPDATE status command

GUI UPDATE command may also be used as a status command to inquire about the panel subunit capability. The following table shows the value of each field in the STATUS command and STABLE and REJECTED frame.

Table 9.9 – Command and response frame for GUI status command

field	STATUS command frame	REJECTED response frame	STABLE response frame
source_plug	not used (FF_{16})	←	←
generation_ID	not used (FF_{16})	←	supported generation_ID value
GUI_resolution	not used (FF_{16})	←	supported GUI_resolution value
scope	not used (FF_{16})	←	←
subfunction	not used (FF_{16})	←	←

Note: "← " means the same value as the previous frame

In case GUI UPDATE status command, source_plug, generation_ID, GUI_resolution, scope and subfunction field shall be set to FF_{16}; and the panel subunit returns the supported value for GUI_resolution and generation_ID field, and the panel subunit doesn't care about source_plug, scope and subfunction field.

If the panel subunit supports this command and doesn't permit the return of the status at the time after receiving the command, it returns the REJECTED response regardless of the owner existence.

If the panel subunit supports only the indirect mode, the NOT IMPLEMENTED response shall be returned anytime for the GUI UPDATE status command.

9.1.3 The source plug behavior

The panel subunit has zero or more source plugs. When the panel subunit supports only the indirect mode, it has no source plug.

In case of the direct model, there is one or more source plugs in the panel subunit. Each source plug can have at most one controller as its owner. So the panel subunit shall reject any control commands sent to the occupied source plug from any other controller than the owner.

When the controller starts working with the panel subunit, the controller can check the panel subunit GUI resolution capability using GUI UPDATE status command but it doesn't need to check this capability if it wants to use only GUI data based on 640 x 480 pixels.

9.1.3.1 Normal procedure

After the controller decides the GUI resolution, it sends GUI UPDATE control command with "open" subfunction to the panel subunit using any available source plug (FF_{16}) in the *source_plug* field.

When the panel subunit accepts the GUI UPDATE command with "open" subfunction, it gives the assigned source plug number and its ownership to the sender of the command. For all other subfunctions, any available source plug (FF_{16}) shall not be used for the value of the *source_plug* field.

After the controller opens the source plug, it makes an internal connection between the panel subunit source plug and the target unit plug, i.e. an asynchronous plug. In this case, it should use CONNECT control command with any available asynchronous plug (BF_{16}) because the internal connection might be permanent.

Afterwards controller establishes the asynchronous connection between the target's asynchronous plug and the controller's using ALLOCATE ATTACH subfunction of ASYNCHRONOUS CONNECTION command specified in [R3].

Note that the connection between the panel subunit and the controller should be established here, so the other way to make the connections could be used in spite of the CONNECT and ALLOCATE ATTACH subfunction of ASYNCHRONOUS CONNECTION.

After all connections are established, the controller can issue the "start" subfunction of the GUI UPDATE command to start GUI update notification on the asynchronous connection with the specified scope value and the assigned source plug number. In case of "current panel" and "add" scope, the trigger of starting GUI update notification is the PUSH GUI DATA control command. Therefore, the panel subunit shall not send any GUI updates to the controller before the first PUSH GUI DATA control command with a panel's element type value.

When the controller wants to stop the GUI update notification, it sends the "stop" subfunction. The controller still owns the source plug even after it sends the "stop" subfunction.

To re-start the interaction with the panel subunit, the controller may invoke another "start" subfunction to the panel subunit after the "stop" subfunction with the source plug number that the controller owns.

The "close" subfunction is used when the controller finishes working with the panel subunit. After the "stop" subfunction is accepted, the controller can break the connection between the asynchronous plugs of the panel subunit and itself. After that, the controller shall issue the "close" subfunction. The panel subunit shall release the ownership of the source plug after it accepts the "close" subfunction command, and then it can accept the "open" subfunction from any controller.

The panel subunit shall reject the "close" subfunction if the controller issues it before the "stop" subfunction is accepted.

The controller should issue a "close" subfunction if it receives the accepted response for an "open" subfunction. Similarly the controller should issue a "stop" subfunction if it receives the accepted response for a "start" subfunction.

9.1.3.2 Bus reset

During ten seconds after a bus reset, the panel subunit shall keep the previous condition and the ownership for the occupied source plug. After ten seconds from the bus reset, if the panel subunit doesn't accept a "restore" subfunction command to the occupied source plug, it shall cancel both the previous condition and the ownership of the source plug, and make the source plug free.

If a bus reset occurs while the controller has ownership and the asynchronous connection is established, the panel subunit shall keep the previous condition for the occupied source plug during the ten seconds. That is, it shall not discard the transmitting data, and keep the internal connection between the source plug and the asynchronous plug.

When the controller wants to continue working with the panel subunit, the controller shall invoke "restore" subfunction to the panel subunit with the owned source plug number within ten seconds after bus reset.

After “restore” subfunction is accepted, the controller sends the ASYNCHRONOUS CONNECTION command with RESTORE PORT subfunction to the target unit. When the RESTORE PORT subfunction is accepted, the controller keeps the ownership of the source plug and the source plug continues to transmit the GUI updates. By using the resume capability of the asynchronous connection, the data is properly transmitted to the controller even if it is interrupted by bus reset. After the above restore procedure succeeds, the panel subunit continues to send the new GUI updates and to accept new PUSH GUI DATA control commands.

Any controller other than the owner shall not send “restore” subfunction to the panel subunit during ten seconds after bus reset.

When another bus reset occurs during the ten seconds period, the panel subunit shall still keep the ownership for the occupied source plug, and the controller shall try to do the above procedure again within ten seconds after the new bus reset if it wants to keep the ownership and connections.

In case the controller has the ownership of the source plug and neither internal connection nor asynchronous connection is established, if the bus reset occurs, the panel subunit shall keep the ownership for the occupied source plug. So the controller shall send “restore” subfunction to the panel subunit during ten seconds if it wants to keep ownership.

When another bus reset occurs during the ten seconds period, the panel subunit shall still keep the ownership for the occupied source plug, and the controller shall send “restore” subfunction to the panel subunit within ten seconds after new bus reset if it wants to keep the ownership.

In case the controller has the ownership of the source plug and it didn’t establish asynchronous connection but it already established the internal connection between the source plug and the asynchronous plug, if the bus reset occurs, the panel subunit shall keep the ownership for the occupied source plug. So it shall send “restore” subfunction to the panel subunit within ten seconds and make internal connection again.

If another bus reset occurs during the ten seconds period, the panel subunit shall still keep the ownership for the occupied source plug, and the controller shall send “restore” subfunction to the panel subunit within ten seconds after new bus reset and then make the internal connection again if the controller wants to keep the ownership and internal connection.

9.1.3.3 Error cases

When the panel subunit receives GUI update command in other order than the specified procedure above, i.e. “start” follows “open” and “close” follows “stop”, the panel subunit shall reject the command.

The controller can retry to send the rejected command to the panel subunit within ten seconds after bus reset.

However the controller should conform to the following cases if it failed to get the accepted response from the panel subunit within the ten seconds.

In case “restore” subfunction failed, the controller shall restart with “open” subfunction after ten seconds from bus reset if it wants to take the ownership of the source plug.

In case RESTORE PORT subfunction of the ASYNCHRONOUS CONNECTION command failed, the controller shall restart with establishing the internal connection between the source plug and the asynchronous plug after ten seconds from bus reset if it wants to resume the asynchronous connection.

9.1.4 GUI update notification data

At anytime, the panel subunit may send panel data to the controller in order to notify the panel data changes after “start” subfunction except that the current panel is not identified. If the current panel is not identified, the updated panel data shall not be transferred to the controller until the next PUSH GUI DATA control command with a panel’s element type value is accepted.

The GUI update data is transmitted on the asynchronous connection and it contains plural GUI elements and a generation number within one frame of asynchronous connection. The GUI elements to be modified by the changes in the target at that time should be put in the same frame and they have the specified generation number. The frame format is described in section 6.2 and 7.2.

The controller may discard the received GUI update data and request not to continue to send this GUI update frame data at that time by transmitting TOSS indication to the target. Note that PUSH GUI DATA command with the “clear” subfunction never suspends any GUI update frame data transmitting on the asynchronous connection.

In case the scope change request is accepted, the panel subunit shall not transfer the GUI updates of the old scope except for the current transmitting frame data.

9.2 PUSH GUI DATA control/status command

The PUSH GUI DATA control/status command allows a controller to request specifically some GUI data from the panel subunit. This command is defined for a CONTROL and a STATUS command. Panel subunit permits the PUSH GUI DATA control command only from the owner of the source plug.

The format of the PUSH GUI DATA control/status command is as follows:

	msb							lsb
opcode	PUSH GUI DATA ($7E_{16}$)							
operand[0]	source_plug							
operand[1]	subfunction							
operand[2]	generation_number							
operand[3]								
operand[4]	status							
operand[5]	indicator							
operand[6]	element_identifier							
operand[7]								
operand[8]								
operand[9]								

Figure 9.2 – PUSH GUI DATA command frame

The *source_plug* field specifies the source plug number, and this source plug will output the GUI data specified in the indicator and element_ID field. If the controller is not the owner of this source plug, then the panel subunit shall return the REJECTED response for the PUSH GUI DATA command.

The *subfunction* field describes the subfunctions of this command.

The following table contains the PUSH GUI DATA subfunction definitions.

Table 9.10 – Subfunction values

subfunction value	subfunction	comments
00_{16}	new	Panel subunit starts to send the specified GUI data
01_{16}	clear	Panel subunit cancels to send the specified GUI data
all other values		Reserved for future definition

When the controller wants to get GUI data via the asynchronous connection, the “new” subfunction shall be used. When the panel subunit accepts a PUSH GUI DATA with “clear” subfunction, it shall no longer transfer the GUI element specified by the element_identifier in that command. However, this “clear” subfunction affects only the “new” subfunction accepted just before itself.

When the element_identifier in the PUSH GUI DATA command indicates a GUI data out of the specified notification scope, the command shall be rejected with “no element” in the status field.

The *generation_number* field describes the current GUI generation of the controller’s scope. When the controller issues this command, the *generation_number* field shall be set to FF_{16} and will be returned with the current generation_number. If the returned *generation_number* is 00_{16}, this panel subunit will not change any GUI element in this panel.

The *status* field describes the current status of the panel subunit. When the controller issues this command, the *status* field shall be set to FF_{16}. Then the panel subunit returns the current status of itself.

The following table contains the status definitions in case of PUSH GUI DATA control command.

Table 9.11 – Status values for PUSH GUI DATA control command

status value	status	comments
00_{16}	no error	The command is accepted with no error.
81_{16}	preparation	Used for “new” subfunction. The panel subunit is preparing to send a GUI data to the source plug, i.e. source plug is reserved.
82_{16}	source plug busy	Used for “new” subfunction. The source plug is transferring a GUI data.
83_{16}	no element	Used for “new” subfunction. The specified data is not in the current scope.
84_{16}	not connected	Used for “new” subfunction. The specified source plug is not connected to any Asynchronous plug.
85_{16}	not owner	The controller is not owner of the specified source plug.
86_{16}	canceled	Used for “new” subfunction. The specified data transmission is canceled halfway.
87_{16}	not transmitting	Used for “clear” subfunction. The specified data is in neither preparation nor transmitting.
FE_{16}	any other error	The other internal error of the panel subunit.
all other values		Reserved for future definition

Note that the status field focuses only on the source plug in the panel subunit, so the panel subunit doesn’t cover any transmission from the panel subunit source plug to the target asynchronous plug and the asynchronous connection between the target and the controller.

The following table contains the status definitions in case of PUSH GUI DATA status command.

Table 9.12 – Status values for PUSH GUI DATA status command

status value	status	comments
01_{16}	ready	The panel subunit is ready status to send a GUI data
81_{16}	preparation	The panel subunit is preparing to send a GUI data to the source plug, i.e. source plug is reserved.
82_{16}	source plug busy	The source plug is transferring a panel data.
84_{16}	not connected	The specified source plug is not connected to any asynchronous plug.
86_{16}	cancel	The specified data transmission is canceled halfway.
FE_{16}	any other error	The other internal error of the panel subunit.
all other values		Reserved for future definition

Note that the status field focuses only on the source plug in the panel subunit, so the panel subunit doesn't cover any transmission from the panel subunit source plug to the target asynchronous plug and the asynchronous connection between the target and the controller.

The *element_identifier* field contains an identifier that uniquely identifies any element within the panel data structure i.e. Panel, Group, GUI element and Data entity, and the *element_type* field in the *element_identifier* is specified in section 7.1.

The panel subunit shall reject the PUSH GUI DATA control command, if the specified data by the *element_identifier* field is out of the current scope and is not a panel, a CONTENT ICON, a DEVICE ICON or a VENDOR ICON.

If the *element_type* is a panel, i.e. 0000_{16} and the *element_ID* is $FFFF_{16}$, then the *element_identifier* indicates the root panel. In this case, the panel subunit shall modify this value to the *element_identifier* value of the root panel and return it to the controller. The panel subunit will transfer the root panel and its element according to the *indicator* value.

If the *element_type* in *element_identifier* field is a DEVICE ICON, i.e.0041_{16} and the *element_ID* is $FFFF_{16}$, then it means a request for a DEVICE ICON, so the panel subunit transmits the DEVICE ICON and its data entities according to the *with_data* bit in the *indicator* value. The request of "device icon" is permitted whenever the controller wants the DEVICE ICON and/or linked data entities, i.e. device bitmap data and device name text data, after the connection is established. The updates of the DEVICE ICON can be transmitted regardless of notification scope. The structure of DEVICE ICON is specified in section 10.5.15.

If the *element_type* in *element_identifier* field is a VENDOR ICON, i.e.0042_{16} and the *element_ID* is $FFFF_{16}$, then it means a request for a VENDOR ICON, so the panel subunit transmits the VENDOR ICON and its data entities according to the *with_data* bit in the *indicator* value. The request of "vendor icon" is permitted whenever the controller wants the VENDOR ICON and/or linked data entities, i.e. vendor bitmap data and vendor name text data, after the connection is established. The structure of VENDOR ICON is specified in section 10.5.16.

If the *element_type* in *element_identifier* field is a CONTENT ICON, i.e.0040_{16} and the *element_ID* is $FFFF_{16}$, then it means a request for CONTENT ICONs, so the panel subunit transmits all CONTENT ICONs managed by the panel subunit at that time according to the *with_data* bit in the *indicator* value. In

this case, it is recommended that the controller subscribe GUI updates for “content icons” notification scope, so the controller can get the updates of the CONTENT ICONs.

On the other hand, the controller can get the CONTENT ICONs in the panel within the current notification scope using PUSH GUI DATA control command, but the element_ID value shall not be $FFFF_{16}$.

The structure of CONTENT ICON is specified in section 10.5.14.

The *indicator* field describes the requested GUI element(s) with element_identifier field.

The format of the indicator field is specified as follows:

	msb							lsb
operand[5]	reserved						level	with_data

Figure 9.3 – Indicator field in PUSH GUI DATA command frame

The *with_data* bit specifies whether or not the data entities, directly linked from the specified element, are requested with the specified element(s). When the *with_data* bit is set to one, the panel subunit sends the directly linked data entities to the controller with the specified element(s). When the *with_data* bit is set to zero, the panel subunit sends no data entity with the specified element(s).

The *level* field specifies the one or plural GUI elements that the controller requests. The following table contains the *level* definitions for panel subunit. When the data entity is requested, the *level* field shall be set to 00_2 and the *with_data* bit shall be set to one.

Table 9.13 – Level values

level value	range	comments
00_2	itself	Only the specified element itself
01_2	itself and next level	The specified GUI element and all GUI elements directly linked from it.
10_2	all level	All GUI elements in a panel
11_2		Reserved for future extension

The panel has a hierarchical structure that is specified by the element_identifier of the panel, group, GUI element and data entity. A panel and its associated group(s), GUI element(s) and data entities, make up the scope of “current panel”.

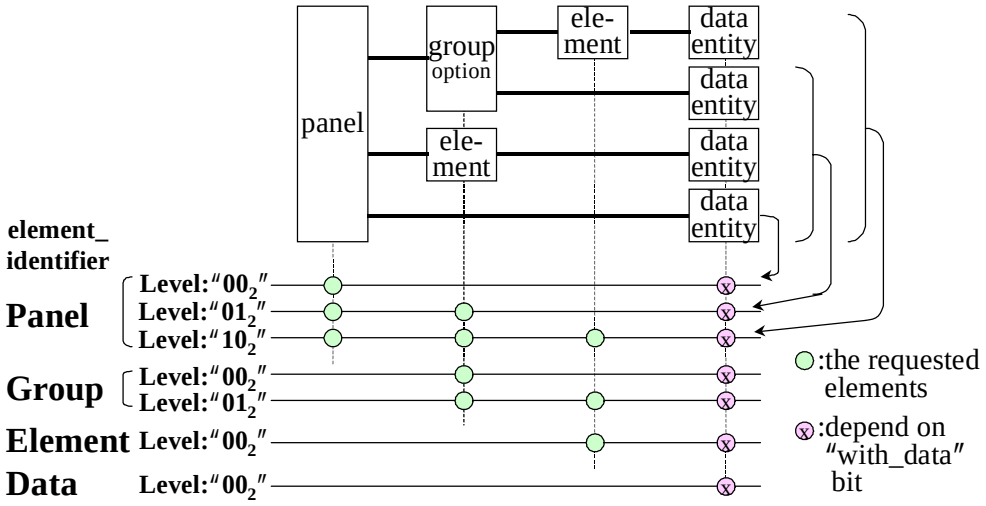

Figure 9.4 – Panel data hierarchy

For example;

When the "level" is 00_2, "with_data" is zero and element_identifier indicates a panel, then *indicator* field specifies the panel itself.

When the "level" is 00_2, "with_data" is one and element_identifier indicates a panel, then *indicator* field specifies the panel and the directly linked data entities from it.

When the "level" is 01_2, "with_data" is one and element_identifier indicates a panel, then *indicator* field specifies the panel, groups, the directly linked GUI elements from the panel and the directly linked data entities from the panel or groups.

When the "level" is 10_2, "with_data" is one and element_identifier indicates a panel, then *indicator* field specifies all GUI element in the panel, i.e. the panel, groups, all GUI elements and all data entities in the panel.

When the "level" is 01_2, "with_data" is one and element_identifier indicates a group, then *indicator* field specifies all GUI elements in the group, i.e. the group and the directly linked GUI elements from the group and the directly linked data entities from the groups or GUI element.

If the element_identifier indicates a GUI element or a data entity, then the "level" field shall be set to 00_2.

If the element_identifier indicates a group, then the "level" field shall be set to either 00_2 or 01_2.

The panel subunit shall transmit the element, if there is at least one suitable element to the condition that *indicator* and *element_identifier* fields indicate. If not, the panel subunit shall send back the REJECTED response with "no element" status.

The panel subunit shall send all elements specified in a PUSH GUI DATA control command by using one frame and the transmission order of the element in the frame should be the hierarchy order; the higher element in hierarchy shall be transferred before the lower one. In the same level, the transmission order should be based on the entry order in the organizational element.

9.2.1 PUSH GUI DATA control command

PUSH GUI DATA command may be used as a control command to request a panel data transmission on the asynchronous connection from the source plug, so the controller shall specify the source plug number at least.

The following table shows the value of each field in the CONTROL command and ACCEPTED and REJECTED frame.

Table 9.14 – Command and response frame for PUSH GUI DATA control command

field	CONTROL command frame	REJECTED response frame	INTERIM response frame	ACCEPTED response frame
source_plug	specified source plug number	←	←	←
subfunction	specified subfunction	←	←	←
generation_number	not used ($FFFF_{16}$)	←	←	current generation _number value
status	not used (FF_{16})	current status value	not used (FF_{16})	no error (00_{16})
indicator	specified indicator value	←	←	←
element_identifier	specified element_identifier value, or specified element_type (XXXX $FFFF_{16}$)	←	←	specified element_identifier value

Note: "← " means the same value as the previous frame

"XXXX" is element_type value of DEVICE ICON, VENDOR ICON, CONTENT ICON or PANEL

When panel subunit supports this command and doesn't permit execution of the command at that time, it shall return the REJECTED response with proper status information.

The following table contains the valid status definitions in case of "new" and "clear" subfunctions in PUSH GUI DATA control command.

Table 9.15 – Status field values for PUSH GUI DATA control command

status value	status	"new" subfunction	"clear" subfunction	comments
00_{16}	no error	ACCEPTED response frame	ACCEPTED response frame	No error for the request
81_{16}	preparation	REJECTED response frame	not used	Panel subunit is preparing the other data
82_{16}	source plug busy	REJECTED response frame	not used	Panel subunit is transmitting the other data on the plug
83_{16}	no element	REJECTED response frame	not used	The specified data is out of the current notification scope.
84_{16}	not connected	REJECTED response frame	not used	Internal and/or asynchronous connection is not connected
85_{16}	not owner	REJECTED response frame	REJECTED response frame	The sender is not owner of the specified source plug
86_{16}	canceled	REJECTED response frame	not used	The specified data transmission is canceled halfway
87_{16}	not transmitting	not used	REJECTED response frame	There is no transmitting data from the specified source plug
FE_{16}	any other error	REJECTED response frame	REJECTED response frame	The request is rejected by any other reason
All other values		Reserved for future definition	Reserved for future definition	

In "new" subfunction, the priority among "not owner", "not connected", "no element", "source plug busy", "preparation" and "any other error" is,

"not owner">"not connected">"no element">"source plug busy" or "preparation">"any other error".

A "canceled" is used when the command is canceled by the new PUSH GUI DATA command with "clear" subfunction.

In "clear" subfunction, the priority among "not owner", "not transmitting" and "any other error" is,

"not owner">"not transmitting">"any other error"

So the panel subunit shall not send the information of low priority error information among these values when the superior error information is existed, i.e. if the panel subunit has both "not owner" error and "not connected" error, then the status field in the rejected response frame shows "not owner".

The "new" subfunction is used for the request of arbitrary set of GUI elements, and this subfunction is valid after the panel subunit accepts and executes "start" subfunction of GUI UPDATE command. However, if the requested GUI element is out of the scope of the last GUI UPDATE command, then this "new" subfunction is rejected with "no element" status. By receiving this subfunction, the panel subunit shall not change the notification scope. The current panel shall be changed only when the panel subunit accepts the request of a panel with "new" subfunction in PUSH GUI DATA control command.

The "clear" subfunction is used for the cancel of the just previously requested GUI elements, and this subfunction is valid after the panel subunit accepts and executes "new" subfunction of PUSH GUI DATA command. All operands of the PUSH GUI DATA with the "clear" subfunction shall be set to the same value as the previous the PUSH GUI DATA with "new" subfunction except subfunction field. If not, then the panel subunit shall reject the PUSH GUI DATA with "clear" subfunction. In the "clear" subfunction the controller shall not specify only some of the elements that are specified in the previous "new" subfunction.

The PUSH GUI DATA control command is valid only for the owner of the source plug, which is connected to the asynchronous plug of the unit connected to the controller. However, the panel subunit doesn't care about the transmission on the asynchronous connection and the internal connection between the source plug and the asynchronous plug, so the controller shall check the received data when the transferred data has error or no data is transferred. When the controller check the status of the panel subunit, the PUSH GUI DATA status command is useful for the check of the data transfer from source plug. When the controller check the internal connection, the CONNECT status command is useful.

The following figure shows the available timing of PUSH GUI DATA control command.

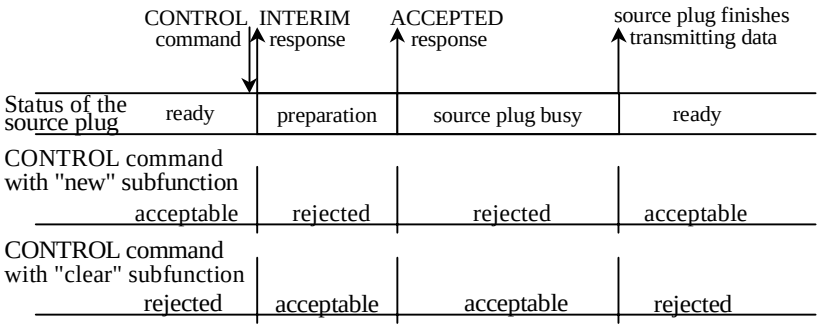

Figure 9.5 – Acceptable timing for the next PUSH GUI DATA command

Panel subunit may or may not return the INTERIM response in the response of PUSH GUI DATA control command with the "new" subfunction. The panel subunit shall reject the PUSH GUI DATA control command with the "new" subfunction while it is preparing the specified data and output it to the source plug.

On the other hand, Panel subunit may accept the PUSH GUI DATA control command with the "clear" subfunction while it is preparing the specified data requested by the previous the PUSH GUI DATA control command with the "new" subfunction and output it to the source plug.

The controller can notice whether the panel subunit finish to send the specified panel data by using LAST indication from asynchronous connection.

Note that the INTERIM response for the "new" subfunction is optional.

The following figure shows the timing of PUSH GUI DATA control command when the panel subunit stops sending the panel data.

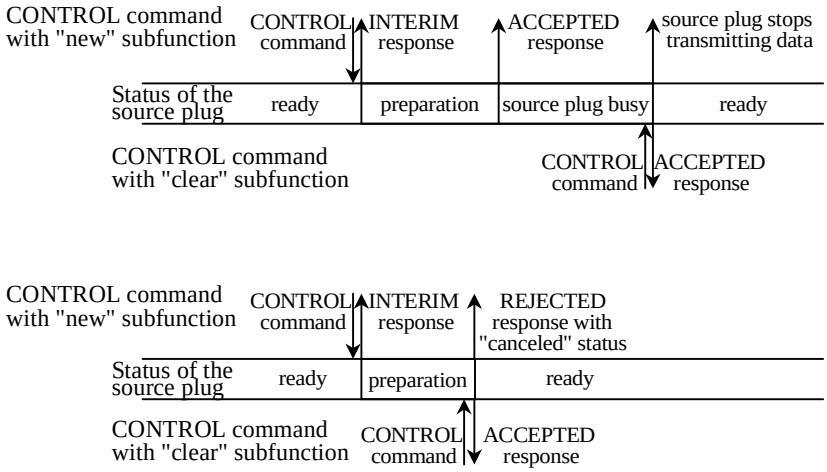

Figure 9.6 – Panel subunit behavior for “clear” subfunction

When the panel subunit receives the PUSH GUI DATA control command with “clear” subfunction, the panel subunit immediately stops transmitting of the panel data which it is preparing or sending.

When the panel subunit is in the source plug busy status, i.e. it is sending the requested data, if it receives the “clear” subfunction, then the panel subunit returns the ACCEPTED response if possible and stops transmitting the data. The controller notices the cancel of the data transmission by LESS indication from asynchronous connection.

When the panel subunit is in the preparation status, i.e. it is preparing the requested data, if it receives the “clear” subfunction, then the panel subunit returns the ACCEPTED response for the “clear” subfunction if possible and stop preparing the data. So the requested data is not transferred at all. The panel subunit returns the REJECTED response for “new” subfunction, so the controller notices the cancel of “new” subfunction.

If the “clear” subfunction is rejected by any reason, the panel subunit continues to transmit the specified data.

In case of the “clear” subfunction, the panel subunit shall not send the INTERIM response.

9.2.2 PUSH GUI DATA status command

PUSH GUI DATA command may also be used as a status command to inquire the information about status of the source plug, e.g. which element the source plug is transmitting, so the controller shall specify the source plug number at least when using this status command.

The PUSH GUI DATA status command from any controller should be accepted anytime.

The following table shows the value of each field in the STATUS command and STABLE and REJECTED frame.

Table 9.16 – Command and response frame for PUSH GUI DATA status command

field	CONTROL command frame	REJECTED response frame	STABLE response frame
source_plug	specified source plug number	←	←
subfunction	not used (FF_{16})	←	←
generation_number	not used ($FFFF_{16}$)	←	current generation _number value
status	not used (FF_{16})	←	current status value
indicator	not used (FF_{16})	←	current indicator value if status field value is preparation or source plug busy, or no information (FF_{16})
element_identifier	not used ($FFFF\ FFFF_{16}$)	←	current element_ identifier value if status field value is preparation or source plug busy, or no information ($FFFF\ FFFF_{16}$)

Note: "← " means the same value as the previous frame

In case PUSH GUI DATA status command, generation_number shall be set to $FFFF_{16}$and element_identifier shall be set to $FFFF\ FFFF_{16}$, status and indicator shall be set to FF_{16} then the panel subunit returns the current value for each field. Further more the subfunction field shall be set to $FF_{16,}$ and the panel subunit doesn't care about this subfunction field.

When panel subunit supports this command and doesn't permit the return of the status at that time, it returns the REJECTED response.

The following table contains the valid status definitions in case of PUSH GUI DATA status command.

Table 9.17 – Available status field value for PUSH GUI DATA status command

status value	status	response	comments
01_{16}	ready	STABLE response frame	The source plug is ready for the request
81_{16}	preparation	STABLE response frame	Panel subunit is preparing the other data
82_{16}	source plug busy	STABLE response frame	Panel subunit is transmitting the other data on the plug
84_{16}	not connected	STABLE response frame	Internal and/or asynchronous connection is not connected
All other values		Reserved for future definition	

The priority among "not connected", "source plug busy" and "preparation" is,

"not connected" > "source plug busy" > "preparation".

So the panel subunit shall not send the information of low priority error information among these values when the superior error information is present.

If the source plug is not connected to any target unit plug, the panel subunit returns the stable response with "not connected" status, and the indicator and element_identifier fields have no information.

If the specified source plug doesn't have an owner, then the panel subunit returns the "ready" status for the PUSH GUI DATA status command.

9.3 USER ACTION control command

The USER ACTION control command is used to inform the target that the user has manipulated an on-screen GUI element. ONLY the owner of the source plug can send USER ACTION control command. The USER ACTION control command from any other controller shall be rejected. As for the detail of the owner, refer to section 9.1.

The control command has the following format:

	msb							lsb
opcode	USER ACTION ($7F_{16}$)							
operand[0]	source_plug							
operand[1]	generation_number							
operand[2]								
operand[3]	element_identifier							
operand[4]								
operand[5]								
operand[6]								
operand[7]	action_type							
operand[8]	action_type_dependent_data							
:								
:								

Figure 9.7 – USER ACTION command format

The *source_plug* field specifies the source plug number, and this source plug outputs the GUI data.

The *generation_number* field describes the GUI generation of the controller's scope. The controller will send USER ACTION command with the latest generation number.

The *element_identifier* field contains an identifier that uniquely identifies the GUI element within the panel data structure. The *element_type* field in the *element_identifier* shall be one of the GUI_element_type values that are described with valid user actions specified in Table 8.2.

The *action_type* field specifies the type of actions for the GUI element and contains the new value, requested by the controller, for the GUI element. The permitted *action_type* field value depends on the *element_type* of the GUI element, as explained in section 8.1.2.

The *action_type_dependent_data* field contains the data specified by each action type.

The following table contains the *action_type* definitions for the GUI elements.

Table 9.18 – Action type values

action type	action_type value	action_type_ dependent_data field	comments
select	00_{16}	not used	The select action by user. No action_type_dependent_data field
press	01_{16}	not used	The press action by user. No action_type_dependent_data field
release	02_{16}	not used	The release action by user. No action_type_dependent_data field
set_value	03_{16}	sixteen bits unsigned integer value	The set value by usr input. The action_data_length field value is two.
enter_data	04_{16}	character strings	The entered value by usr input. The action_data_length field value is dependent to user input.
choose_list	05_{16}	index list	A list of index number of the chosen elements The action_data_length field value is dependent to user input.
select_item	06_{16}	index number	An index number of the selected state The action_data_length field value is one.
select_element	07_{16}	element number	A sequence number of the selected element The action_data_length field value is one.
reserved	all other values		Reserved for future definition

The *select* action is transmitted when the user selects the GUI element.

The *press* action is transmitted when the user presses the button.

The *release* action is transmitted when the user releases the button.

The *set_value* action is transmitted when the user sets the value to the SHOW RANGE or SET RANGE GUI element, e.g. slider or dial. The action_data field value (Xr) is unsigned sixteen bits integer value.

zero < = Xr <= value range attribute value.

The *enter_data* action is transmitted when the user input the data to the ENTRY GUI element. The action_data field value is defined in the section 10.5.8.

The *choose_list* action is transmitted when the user chooses the candidates in CHOICE GUI element. The action_data field value is a list of index number of the chosen element(s), and each index list (Xc) is unsigned eight bits number.

one < = Xc <= the number of candidates

The *select_item* action is transmitted when the user selects the state in TOGGLE GUI element. The action_data field value (Xt) is unsigned eight bits number.

one < = Xt <= the number of states

The *select_element* action is transmitted when the user selects the element in ANIMATION GUI element. The action_data field value (Xa) is unsigned eight bits number.

one < = Xa <= the number of elements

If the animation type is once playback, i.e. repetition field value is zero, and the controller finishes the playback without user action, then it shall send USER ACTION control command that has the special action _data field value (FF_{16}).

Table 9.19 – Action types

action type	action_type field value	PANEL	GROUP	PANEL LINK	BUTTON	ANIMATION	SHOW RANGE	SET RANGE	ENTRY	CHOICE	TEXT	STATUS	ICON	TOGGLE	CONTENT ICON	DEVICE ICON	VENDOR ICON
select	00_{16}	-	-	X	-	-	X	-	-	-	X	-	X	-	X	-	-
press	01_{16}	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-
release	02_{16}	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-
set_value	03_{16}	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-
enter_data	04_{16}	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
choose_list	05_{16}	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-
select_item	06_{16}	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-
select_element	07_{16}	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
Reserved	all other value																

Note that “X” means “available” for the GUI element and “-” means “not used” for the GUI element.

The format of the action_type_dependent_data field is specified below,

address offset	msb							lsb
operand[8]	remaining_blocks							(msb)
operand[9]	action_data_length							(lsb)
operand[10]	action_data							
:								

Figure 9.8 – Action_type_dependent_data field in the USER ACTION command format

The *remaining_blocks* specifies the number of the remaining command frames to send the information of the user action, and the value of this field shall be zero if the action_type is neither enter_data nor choose_list.

In case the action_type is enter_data or choose_list, the controller can determine the size of action_data field, which target device can receive, using the “action_data_size” optional attribute of the ENTRY or

CHOICE GUI element. If the element doesn't have "action_data_size" optional attribute or there is neither ENTRY nor CHOICE GUI element in the panel, the controller should send the action_data in one frame.

This field is used when a command is larger than the maximum command frame size that the target device can receive. When this occurs, the data field is fragmented into N blocks that are sent sequentially, each in one of N separate commands, where each command is small enough to be accommodated by the target device's command buffer. At a minimum, the buffer must be able to hold a command with at least a 32-byte action_data field. The size of the action_data field in the first N-1 fragments shall be the same size and a multiple of 16 bytes greater than or equal to 32 bytes.

Each of the N command frames is identical except for the values in the action_data fields. For the first command, the remaining_blocks field is set to the value of N-1. In each successive command, the remaining_blocks field value is decreased by one until it reaches zero, indicating the last command fragment. If the first fragmented command is rejected, the controller shall not send the remaining fragmented commands.

Since the size of a command or a response frames cannot exceed the 512-byte limit imposed by the underlying FCP transport, the case where a command must be fragmented occurs when a target device has a command frame buffer capacity less than 512 bytes. Typically the command's size is within the target device's command frame buffer capacity and the command is sent without fragmentation and with a remaining_blocks value of zero.

The action_data_length field specifies the length of action_data field in bytes.

The action_data field contains the data to be transferred, and the contents of the data field depend on the action_type.

In case the action_type is set_value, the remaining_blocks field shall be set to zero, action_data_length field to one and the action_data field be one byte value.

The response of this command uses the AV/C response frame fully explained in reference [R1].

The REJECTED and ACCEPTED response frames shall not contain the action_data field.

The following table shows the value of each field in the CONTROL command, ACCEPTED response and REJECTED response.

Table 9.20 – Command and response frame for USER ACTION control command

field	CONTROL command frame	REJECTED response frame	ACCEPTED response frame
source_plug	specified source plug number	←	←
generation_number	specified generation _number value	←	←
element_identifier	specified element identifier	←	←
action_type	specified action type	←	←
data	specified action information	not used (not transferred)	←

Note: "← " means the same value as the previous frame

In the ACCEPTED response and the REJECTED response, the data field, which is transmitted in CONTROL command, shall not be transferred.

If the controller has not previously invoked a panel GUI update request for the scope to include GUI elements being manipulated by the USER ACTION command, then the USER ACTION command shall be REJECTED.

When the sender of the USER ACTION CONTROL command is not owner of the specified source plug, the panel subunit shall reject this command. If there is no owner of the specified source plug, then the CONTROL command shall be REJECTED.

If the encoded action_type value for user action is not suitable for the specified GUI element by the element_identifier field, then the USER ACTION command shall be REJECTED. The acceptable user actions for the GUI element are defined in Table 8.2.

When the GUI elements have changed as a direct result of this USER ACTION command, the panel subunit transfers the data of the changed GUI elements via asynchronous connection with a new generation number.

When the generation number in CONTROL command is not same as the current generation number in the panel subunit, the panel subunit may or may not return ACCEPTED response, this behavior is completely up to the panel subunit implementation.

9.4 PASS THROUGH control command

The PASS THROUGH command is used to convey the proper user operation to the target (transparently to the user). The PASS THROUGH command can be transferred regardless of the state of the target.

The format of the PASS THROUGH control command is as follows.

	msb							lsb
opcode	PASS THROUGH ($7C_{16}$)							
operand[0]	state_flag	operation_id						
operand[1]	operation_data_field_lengh							
operand[2]	operation_data (operation_id dependent)							
:								

Figure 9.9 – PASS THROUGH command format

The *state_flag* field indicates the user operation of 'pressing or releasing a button', as the remote commander operation. When a button is pressed, the value of this field shall be zero; when released, the value shall be one. A controller usually sends this command twice, i.e. press and release, when user presses and then releases a button. A command with the pressed value is valid for two seconds from the time when a target sends back a response of the command. The controller shall continue sending pressed value with identical operation id value in the *operation_id* field while the command is wished to stay valid. Either if the target has not received the pressed command within two seconds or the target receives the pressed command with another operation id, then the target regards that the released command was sent but missed to receive. In these cases, the target will ignore the released command when the target receives this released command after the time out or reception of the new pressed command.

The "press and hold" user operation could be realized by using press and release operation above. This "press and hold" user operation might mean the different operation from the "press" user operation to the same button. To help the target to decide whether a pressed button is hold or not, it is recommended that the controller have capability of issuing a "release" command within 300 ms, after it receives a response to the last "press" command. The target should not judge the user operation to be "press and hold" within this 300ms.

The *operation_id* field shows the user operation by each operation id value. See the following table.

Table 9.21 – Operation id List

operation_id	user operation		operation_id	user operation
00_{16}	select		35_{16}	display information
01_{16}	up		36_{16}	help
02_{16}	down		37_{16}	page up
03_{16}	left		38_{16}	page down
04_{16}	right		39_{16}	reserved
05_{16}	right-up		:	:
06_{16}	right-down		$3F_{16}$	:
07_{16}	left-up		40_{16}	power
08_{16}	left-down		41_{16}	volume up
09_{16}	root menu		42_{16}	volume down
$0A_{16}$	setup menu		43_{16}	mute
$0B_{16}$	contents menu		44_{16}	play
$0C_{16}$	favorite menu		45_{16}	stop
$0D_{16}$	exit		46_{16}	pause
$0E_{16}$	reserved		47_{16}	record
:	:		48_{16}	rewind
$1F_{16}$	:		49_{16}	fast forward
20_{16}	0		$4A_{16}$	eject
21_{16}	1		$4B_{16}$	forward
22_{16}	2		$4C_{16}$	backward
23_{16}	3		$4D_{16}$	reserved
24_{16}	4		:	:
25_{16}	5		$4F_{16}$	:
26_{16}	6		50_{16}	angle
27_{16}	7		51_{16}	subpicture
28_{16}	8		52_{16}	reserved
29_{16}	9		:	:
$2A_{16}$	dot		70_{16}	:
$2B_{16}$	enter		71_{16}	F1
$2C_{16}$	clear		72_{16}	F2
$2D_{16}$	reserved		73_{16}	F3
:	:		74_{16}	F4
$2F_{16}$	:		75_{16}	F5
30_{16}	channel up		76_{16}	reserved
31_{16}	channel down		:	:
32_{16}	previous channel		$7D_{16}$	:
33_{16}	sound select		$7E_{16}$	vendor unique
34_{16}	input select		$7F_{16}$	reserved

Description of user operation

Notes:

- The indications on a remote commander or other input device can be different from those shown below as “user operation”. Note to use words or symbols that reminds users the targets' performance explained below.
- “Expected operation to be performed by a target” described below is the recommended basic behavior, and does not limit the possibilities to apply these commands for other purposes.

Table 9.22 – Cursor navigation and Menu operations

user operation	Expected operation to be performed by a target
select	Selects the item focused by cursor.
up	Moves cursor upper direction.
down	Moves cursor lower direction.
left	Moves cursor left direction.
right	Moves cursor right direction.
right-up	Moves cursor upper-right direction.
right-down	Moves cursor lower-right direction.
left-up	Moves cursor upper-left direction.
left-down	Moves cursor lower-left direction.
root menu	Displays initial menu to start GUI operation. The menu displayed with this command is target-dependent, so it might be contents menu, setup menu, favorite menu or the other menu, furthermore it may be changed dynamically according to the status of the target. This command may be used to finish GUI operation alternately.
setup menu	Displays set up menu such as option set up. (Can be used as a shortcut.) The menu displayed with this command should be designed to be reached from the initial menu of the target.
contents menu	Displays contents menu. (Can be used as a shortcut.) For example, this command may be used to display the Electric Program Guide (EPG) or the contents list in a storage medium. The menu displayed with this command should be designed to be reached from the initial menu of the target.
favorite menu	Displays user preset menu. (Can be used as a shortcut.) For example, this command may be used to display the list of user preset channel. The menu displayed with this command should be designed to be reached from the initial menu of the target.
exit	Closes current menu and go back previous menu. This command may also be used to finish GUI operation, but a target shall be implemented to be finished GUI operation without this command.

A controller which supports GUI operation shall be implemented with user operations of *select*, *up*, *down*, *left*, *right*, and *root menu*. The GUI of a target should be designed to be controlled all of its available functions by these user operations, if the target doesn't support the direct mode.

Table 9.23 – Numerical input operations

User operation	Expected operation to be performed by a target
0 – 9	Input a numerical value.
dot	Used with 0-9 to input numerical value such as a sub channel in US.
enter	Fix the entered numerical value. Target should be implemented to fix the entered value in any way without this command, such as time out.
clear	Cancel the entered numerical value.

Table 9.24 – Other operations

User operation	Expected operation to be performed by a target
Channel up	Switches the channel, such as broadcast channel, to upper one, i.e. plus direction in number.
Channel down	Switches the channel, such as broadcast channel, to lower one, i.e. minus direction in number.
previous channel	Switches to the previously selected channel. For example, in case 123 ch was switched to 246 ch, this command can be used as a switcher between 123 ch and 246 ch.
sound select	Used to switch the sound such as multiple language selection.
input select	Used to switch the input signal.
display information	Displays the information about current contents. For example, this command may be used to display the channel number, broadcaster and broadcast time, or recorded date and timecode.
help	Displays help instructions.
page up	Scrolls up the whole screen or part of display.
page down	Scrolls down the whole screen or part of display.
power	Controls the power state of the device alternatively. This command may support to turn the device off only.
volume up	Turns the volume to high.
volume down	Turns the volume to low.
mute	Puts the sound out, and may resume it alternatively or not.
play	Starts playing back the specified content at normal speed.
stop	Stops playing back the content.
pause	Stops playing back the content, and may resume to play it back alternatively.
record	Records the specified stream or content to the specified medium.
rewind	Moves the position toward the beginning of the medium.
fast forward	Moves the position away from the beginning of the medium.
eject	Ejects the medium from the device, and may close the door for loading the medium alternatively.
forward	Switches the contents, such as music tune, or video chapter, which can be reproduced with “play” operation, to the forward one. The 'forward' means future direction in time, plus direction in number, and down direction in a list.
backward	Switches the contents, such as music tune, or video chapter, which can be reproduced with “play” operation, to the backward one. The 'backward' means past direction in time, minus direction in number, and up direction in a list.
angle	Switches the scene of the contents, if it has multi angle contents.
subpicture	Switches or rotates the sub pictures, if it has some sub pictures data.

Table 9.25 – Function key operations

User operation	Expected operation to be performed by a target
F1 – F5	Input function keys of F1 – F5

*Operation_id*s for function keys are defined for the purpose of operating actions that are uniquely designed for specific AV device type or a particular device to be controlled by a general purpose controller.

Function key operation is flexibly used according to the type of device or the character of the device, and therefore a particular action of a target is not defined to each function key in this specification. Since function keys are defined for universal use, an industry group may assign a function key to the action that may be commonly shared with the AV devices in the same type. In another case, a device may allow a user to assign his/her favorite action to a function key as short-cut operation. Annex C in this specification describes some examples for function key operation.

The function keys make it possible to control actions that are uniquely assigned by a device from a general purpose controller. A vendor of a device shall present to the user which action is assigned to which function key in some method; such as to describe in the device's users manual, present in OSD, or display on the device.

Table 9.26 – Vendor unique operation

User operation	Expected operation to be performed by a target
vendor unique	Used to convey vendor unique information to a target. Information which can be conveyed with other operation id shall not be handled.

The *operation_data_field_length* field contains the number of bytes in the *operation_data* field. If there is no *operation_data* field, the value of this field shall be zero. In this version of the specification, the *operation_data* field is defined only for the "vendor unique" operation. So the other operation has no *operation_data* field and its *operation_data_field_length* field value is zero in the command frame.

The following table shows the value of each field in CONTROL command, NOT IMPLEMENTED response, ACCEPTED response and REJECTED response. INTERIM response should not be used.

Table 9.27 – Command and response frame for PASS THROUGH control command

field	CONTROL command frame	ACCEPTED response frame	REJECTED response frame	NOT IMPLEMENTED response frame
state_flag	press/release state	←	←	←
operation_id	type of operation	←	←	←
operation_data	depends on operation_id	←	←	←

Note: "← " means the same value as the previous frame

A target sends ACCEPTED response when it is implemented with the requested operation, and sends NOT IMPLEMENTED response when it is not implemented. A target should send ACCEPTED response no matter whether or not the operation can be executed at the moment. A target may send REJECTED response, for example, when it is implemented the requested operation but has been reserved by another controller with RESERVE command.

9.4.1 Operation Data Field format

9.4.1.1 Vendor unique

When the value of *operation_id* field is $7E_{16}$, which means "vendor unique", the *operation_data* field is defined as following table shows.

Address	Contents
00 00$_{16}$	company id
00 01$_{16}$	
00 02$_{16}$	
00 03$_{16}$	vendor dependent information
:	

Figure 9.10 – Operation_data (operation_id = 7E$_{16}$) field format

The *company id* field shows the manufacture that defined the vendor unique user operation. The *company id* value is obtained from the IEEE Registration Authority Committee (RAC). The most significant part of the *company id* is stored in address *0000$_{16}$* and the least significant part in address *0002$_{16}$*.

9.4.2 Controller Implementation guideline for PASS THROUGH command

When the user manipulates the physical or logical remote commander, e.g. IR remote commander or commander that emulates IR remote commander using GUI, the controller may issue PASS THROUGH command or may trap the user operation according to the controller's application.

However implementation guidelines for PASS THROUGH command placed on a controller are as follows:

– A PASS THROUGH command provides a simple and common mechanism to control the target. However, in principle, an application in a controller should not use a PASS THROUGH command as a substitute for any subunit or unit command, because the target behavior to a PASS THROUGH command may not be the same as that controller expects.

– Basically, a PASS THROUGH command intends to be used in the following cases:
(1) When the controller displays the stream data and the user operates a remote commander, the controller should first send the PASS THROUGH command to the source device of the stream data.
(2) When the user selects a target device and operates a remote commander, the controller should send first the PASS THROUGH command to the selected device.

  Afterwards the controller should send the other unit or subunit command when the target returns NOT IMPLEMENTED response, if the controller can surely recognize the other suitable unit or subunit command for the user manipulation.

– Basically, a PASS THROUGH command does NOT intend to be used in the following cases:
(1) When an application of the controller controls plural targets at the same time, e.g. dubbing, the controller should send the unit or subunit command to these targets.
(2) When the controller wants to control a legacy device that doesn't implement PASS THROUGH command, the controller should first send the unit or subunit command to the legacy device.

– Usage in the scenario for the indirect mode;
CONDITION: The controller has a remote commander, whose buttons correspond to user operations defined in Table 9.21, and the target supports the indirect mode.
(1) When the user pushes the button on the remote commander, it should first issue the PASS THROUGH command with the operation_id according to the selected kind of the button.

– Usage in the scenario for the direct mode;
CONDITION: The controller has a remote commander, whose buttons correspond to user operations defined in Table 9.21, and the target supports the direct mode and PASS THROUGH command.
(1) If the focused GUI element has no cursor through attribute, when the user pushes the button

for cursor navigation, i.e. up, down, left, right, right-up, right-down, left-up and left-down, on the remote commander, the controller should not issue any PASS THROUGH command.
(2) If the focus comes into the GUI element whose cursor through attribute is enabled, when the user pushes the button for cursor navigation, i.e. up, down, left, right, right-up, right-down, left-up and left-down, on the remote commander, the controller should issue the PASS THROUGH command.
(3) When the user pushes the menu operation button, i.e. root menu, setup menu, contents menu and favorite menu, on the remote commander, the controller should do starting procedure of the direct mode prior to issuing the PASS THROUGH command if the procedure has not been done. After making connection between the controller and the panel subunit and completing GUI UPDATE start subfunction, the controller should issue the PASS THROUGH command that corresponds to the menu operation button pressed. In case of the “exit” user operation, the controller should issue the PASS THROUGH command whenever it can, and the target should make suggestion to the controller to suggest to finish GUI representation by providing suggested data defined in section 7.8.
(4) When the user pushes the button that means neither cursor navigation focus nor menu operation, the controller should first issue the PASS THROUGH command. This user operation provides short-cut operation.

10. GUI Element Type Definitions

10.1 GUI Element Data Structure Overview

The GUI elements are the structures that a target can use to define how it should be represented on a display. These elements are put on a screen by an application (controller). A given element can be defined to allow the user to interact with it. In these cases, the user can apply a set of actions that result in this element's ID being returned to the target along with the respective action. Some elements exist only to support navigation among elements. In these cases, no actions are sent back to the panel subunit. Navigation can be done between panels, groups, and/or individual elements.

Terminology used throughout this section is defined to be the following:

1) **Navigation**: the controller responding to user inputs for moving from one GUI element to another. How this is done is controller-specific. It could be with a remote control up, down, left, and right arrows, a mouse and keyboard, etc.
2) **Focus**: the result of the controller navigating to a GUI element. This is controller-specific, but the user experience results in the GUI element being highlighted in some manner; this can be by changing its color, rendering a border around its perimeter, etc.
3) **Interactive**: once focus is on a GUI element, the user can place an action on this element resulting in this input being sent to the target device along with the GUI element identifier.
4) **Display-only**: this is a GUI element provided by the target and displayed by the controller. However, the controller does not allow the user to navigate to it, thus not allowing either focus or interaction to take place. This GUI element is for display only.

A panel element is intended to represent an entire screen's worth of GUI elements. Within the panel, there can be zero or more groups of non-organizational elements. In addition to groups, a panel can contain standalone elements that are not part of any group. Navigation can be suggested by the target, for moving from panel to panel, from group to group, and from element to element.

An element consists of it's main definition part, followed by mandatory attributes and optional attributes. The mandatory attributes of a GUI element must be included in the data structure. The optional attributes need not be present but if included are intended to give the display device suggestions for displaying this particular GUI element. These optional attributes are defined in a GUI element format. The collection of valid optional attributes is specified for each GUI element in the detailed specifications below.

All GUI elements shall have a label, which should describe the GUI element function. The label is defined to be a text string of minimum length zero and maximum *suggested* length sixteen. A label is allowed to exceed the maximum suggested length, but controllers are only required to display up to sixteen characters; this might be accomplished by scrolling the characters, in the case of very small LCD text displays. In the case where the label does exceed the maximum suggested length, targets "know" that a user might not be able to see the complete label. GUI designers should take this into account when designing the GUI presentation of the target device.

The concepts of panel, group, and label are used by a controller to present a given panel in one of three modes: full-capability, scaled-down, and text-only. If the target's current panel can be displayed on the screen as suggested by the target, then the display is said to be presented to the user as "full-capability".

If the controller cannot display the entire panel as suggested by the target, it can then fall back to the next mode of "scaled-down" display. Suggestions on how to scale down would be represented by the groups within a panel. The display device could then display one group at a time and move between groups by displaying "next" and "previous" groups. The controller is responsible for this navigation.

The last mode, “text-only”, would be used if the controller can only display text, whether several lines of text at once or only one line of text at a time. In this case, the GUI element label is used to represent the element. The label can be selected by the user for interaction purposes, such as button clicks.

The controller has the freedom to display the GUI elements in any manner it sees fit. The mandatory and optional attributes of elements are intended to be suggestions to the controller on how to organize and present the panels, groups, and standalone elements to the user.

The controller shows the focus in own manner which can be by changing the color of the elements, rendering a border around with it, etc. On the other hand, the GUI element can suggestively give the controller its appearance when it receives the focus or dose not by using some optional attributes or links. Therefore GUI designers can attempt to unite the concept of the appearance when each GUI element receives the focus or dose not in all over the GUI presentation of the target device. When some GUI elements have these optional attributes or links concerning the focus, all of interactive GUI elements in the panel should have those attributes or links.

A GUI element has the attributes “height” and “width”, which define the boundary for the element, and a bitmap for the GUI element shall be smaller than the GUI element size. The GUI element size specifies an area for both bitmap and text label. The attribute “position” is used to place a GUI element in a logical location on the panel relative to all other elements and groups. A panel size will be chosen by the GUI designer to represent a typical screen display. Whatever size is chosen for a panel, all panels in a given target should be the same size; the purpose for this rule is to make life a little easier on controllers when constructing and managing the GUI representation.

A panel is defined to be 2 dimensional with the upper left corner being defined as (x = 0, y = 0). All positions given for groups or standalone elements would be relative to this 0,0 coordinate of the panel. The elements within a group have position values which are relative to the upper left corner of the group; the local upper left corner of the group is x = 0, y = 0. For all structures, the x-coordinate increments from left to right and the y-coordinate increments from top to bottom on the display.

GUI elements can be overlap by their positions. The order of GUI elements in the panel or group indicates the back to front order; The GUI element that have the latest reference in the link fields of the panel or group has the top priority and overlaid on the other GUI elements in the link fields.

Panels and groups can have background colors and/or patterns. A pattern takes the form of a tiled sequence of icons which would allow for textured appearances.

Panels and groups can be associated with audio and/or video content that would be streamed in real-time when the GUI element gets the focus. If it has the capability, the controller can set up audio and video media streams using isochronous channels. Within the optional attribute “audio_video” for panels and groups is the channel number supplied by the target. If the controller can accept media streams then it would listen on this channel number for content being supplied by the target.

A bitmap is an image and a sound is an audio file format. A bitmap and/or sound can optionally be associated with certain GUI elements. Bitmaps and sounds are intended to be relatively small amounts of data in comparison to isochronous audio and video streams being played back in real-time.

To help clarify descriptions, the terms “audio” and “video” will be used when media streams are being discussed, and the terms “bitmap” and “sound” will be used when these shorter, more “static” data types are being discussed.

Sound attributes could be used to attach short sounds to button clicks, releases, and presses. A more complex use of this feature would be to for a panel to have background music while each animation icon could be describing some device function to the user via individual sound clips. Of course, such GUI presentations depend upon the capability of the controller to support multiple sound sequences (the support

of any sound playback is optional for controllers). When these capabilities do not exist, then there will either be individual sounds (not layers) or no sound at all.

10.2 Supporting Data Structures

This section defines some basic structures and definitions which are used in the GUI element specifications.

The GUI element shall have mandatory information specified in each GUI element, and may or may not have optional information. Mandatory or optional information is respectively classified into attributes and link information.

The attributes specify property of each GUI element using the format defined in each attribute. The link information indicates the reference to the other element or data entity defied in section 8.2.1 using element identifier that contains *element_ID* and *element_type*.

10.2.1 Mandatory information for GUI elements

There are two types of mandatory information that is included in the GUI element data structures; mandatory attributes and mandatory links.

The mandatory attributes specify a property of the GUI element and the property is specified by an encoded code, i.e. *info_ type*.

The mandatory links specify a property of the GUI element or data entity, and the property is specified by a link to the other GUI element or the data entity using an encoded code, i.e. *info_type*.

These mandatory attributes and links share the information type (*info_type*) values defined in Table 10.1 – Mandatory info_type field value.

The following table defines the info type values of mandatory information in one byte:

Table 10.1 – Mandatory info_type field value

type value	mandatory attribute or link	PANEL	GROUP	PANEL LINK	BUTTON	ANIMATION	SHOW RANGE	SET RANGE	ENTRY	CHOICE	TEXT	STATUS	ICON	TOGGLE	CONTENT ICON	DEVICE ICON	VENDOR ICON
00_{16}	width height	m	m	m	m	m	m	m	m	m	m	m	m	m	m	-	-
01_{16}	label link	m	m	m	-	m	m	m	m	m	-	m	m	m	m	m	m
02_{16}	image link	-	-	m	-	-	-	-	-	-	-	-	m	-	m	m	m
03_{16}	interactive	-	-	m	m	m	m	m	m	m	m	-	m	m	m	-	-
04_{16}	aspect ratio	m	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
05_{16}	linked panel ID	-	-	m	-	-	-	-	-	-	-	-	-	-	-	-	-
06_{16}	released label	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-	-
07_{16}	released bitmap link	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-	-
08_{16}	pressed label	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-	-
09_{16}	pressed bitmap link	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-	-
$0A_{16}$	animation type	-	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-
$0B_{16}$	animation element link	-	-	-	-	m	-	-	-	-	-	-	-	-	-	-	-
$0C_{16}$	range type	-	-	-	-	-	m	m	-	-	-	-	-	-	-	-	-
$0D_{16}$	value set	-	-	-	-	-	m	m	-	-	-	-	-	-	-	-	-
$0E_{16}$	entry type	-	-	-	-	-	-	-	m	-	-	-	-	-	-	-	-
$0F_{16}$	entry type depend	-	-	-	-	-	-	-	m	-	-	-	-	-	-	-	-
10_{16}	choice type	-	-	-	-	-	-	-	-	m	-	-	-	-	-	-	-
11_{16}	choice element link (not chosen)	-	-	-	-	-	-	-	-	m	-	-	-	-	-	-	-
12_{16}	choice element link (chosen)	-	-	-	-	-	-	-	-	m	-	-	-	-	-	-	-
13_{16}	text link	-	-	-	-	-	-	-	-	-	m	-	-	-	-	-	-
14_{16}	status type	-	-	-	-	-	-	-	-	-	-	m	-	-	-	-	-
15_{16}	work status	-	-	-	-	-	-	-	-	-	-	m	-	-	-	-	-
16_{16}	toggle status	-	-	-	-	-	-	-	-	-	-	-	-	m	-	-	-
17_{16}	toggle element link	-	-	-	-	-	-	-	-	-	-	-	-	m	-	-	-
18_{16}	content availability	-	-	-	-	-	-	-	-	-	-	-	-	-	m	-	-
19_{16}	AV plug	-	-	-	-	-	-	-	-	-	-	-	-	-	m	-	-
all others	reserved																

Note that “m” means “mandatory” for the GUI element and “-” means “not used” for the GUI element.

10.2.2 Optional information for GUI elements

There are two types of optional information that may be included in the GUI element data structures; optional attributes and optional links.

The optional attributes specify a property of the GUI element and the property is specified by an encoded code.

The optional links specify a property of the data entity, and the property is specified by a link to the data entity using an encoded code.

These optional attributes and links share the optional information type (*info_type*) defined in Table 10.2 – Optional info_type field value.

The following table defines the info type values of optional information in one byte:

Table 10.2 – Optional info_type field value

type value	optional attribute or link	PANEL	GROUP	PANEL LINK	BUTTON	ANIMATION	SHOW RANGE	SET RANGE	ENTRY	CHOICE	TEXT	STATUS	ICON	TOGGLE	CONTENT ICON	DEVICE ICON	VENDOR ICON
80_{16}	position	o	o	o	o	o	o	o	o	o	o	o	o	o	o	-	-
81_{16}	font size	o	o	o	o	o	o	o	o	o	o	o	o	o	o	-	-
82_{16}	relation	-	o	o	o	o	o	o	o	o	o	o	o	o	o	-	-
83_{16}	focus navigation	-	-	o	o	o	o	o	o	o	o	o	o	o	o	-	-
84_{16}	help panel ID	-	-	o	o	o	o	o	o	o	o	o	o	o	o	-	-
85_{16}	focus sound link	o	-	o	o	o	o	o	o	o	o	o	o	o	o	-	-
86_{16}	select sound link	-	-	-	-	o	o	o	-	-	o	-	o	o	o	-	-
87_{16}	background color	o	o	-	-	-	o	o	o	o	o	-	-	-	-	-	-
88_{16}	background picture link	o	o	-	-	-	o	o	o	o	o	-	-	-	-	-	-
89_{16}	background pattern link	o	o	-	-	-	o	o	o	o	o	-	-	-	-	-	-
$8A_{16}$	audio video	o	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$8B_{16}$	title link	o	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$8C_{16}$	show with parent	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$8D_{16}$	initial focus	o	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$8E_{16}$	press sound link	-	-	-	o	-	-	-	-	-	-	-	-	-	-	-	-
$8F_{16}$	release sound link	-	-	-	o	-	-	-	-	-	-	-	-	-	-	-	-
90_{16}	max label link	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
91_{16}	min label link	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
92_{16}	center label link	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
93_{16}	value offset	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
94_{16}	value power	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
95_{16}	unit label link	-	-	-	-	-	o	o	-	-	-	-	-	-	-	-	-
96_{16}	default text link	-	-	-	-	-	-	-	o	-	-	-	-	-	-	-	-
97_{16}	hotlink	-	-	-	-	-	-	-	-	-	o	-	-	-	-	-	-
98_{16}	playback time	-	-	-	-	-	-	-	-	-	-	-	-	-	o	-	-
99_{16}	recorded date	-	-	-	-	-	-	-	-	-	-	-	-	-	o	-	-
$9A_{16}$	broadcast date	-	-	-	-	-	-	-	-	-	-	-	-	-	o	-	-
$9B_{16}$	vendor dependent info	o	o	o	o	o	o	o	o	o	o	o	o	o	o	-	-
$9C_{16}$	safety area position	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$9D_{16}$	overlay	o	-	-	-	-	o	-	-	-	o	o	o	-	-	-	-
$9E_{16}$	action data size	-	-	-	-	-	-	-	o	o	-	-	-	-	-	-	-
$9F_{16}$	launch bitmap link	-	-	-	-	-	-	o	o	o	-	-	-	-	-	-	-

continue to next page

Note that “o” means “optional” for the GUI element and “-” means “not used” for the GUI element.

Table 10.2 – Optional info_type field value (cont.)

type value	optional attribute or link	PANEL	GROUP	PANEL LINK	BUTTON	ANIMATION	SHOW RANGE	SET RANGE	ENTRY	CHOICE	TEXT	STATUS	ICON	TOGGLE	CONTENT ICON	DEVICE ICON	VENDOR ICON
continuation of the previous page																	
$A0_{16}$	AV plug info	-	-	-	o	-	-	o	o	o	o	-	o	-	-	-	-
$A1_{16}$	cursor through	-	-	-	-	-	-	-	-	-	-	-	o	-	-	-	-
$A2_{16}$	assured area size	-	o	-	-	-	-	-	-	-	-	-	o	-	-	-	-
$A3_{16}$	focused label link	-	-	o	o	o	-	-	-	-	-	-	o	o	o	-	-
$A4_{16}$	focused bitmap link	-	-	o	o	o	-	-	-	-	-	-	o	o	o	-	-
$A5_{16}$	background focused color	-	-	-	-	-	o	o	o	o	o	-	-	-	-	-	-
$A6_{16}$	background focused picture link	-	-	-	-	-	o	o	o	o	o	-	-	-	-	-	-
$A7_{16}$	background focused pattern link	-	-	-	-	-	o	o	o	o	o	-	-	-	-	-	-
$A8_{16}$	data entity size	o	o	o	o	o	o	o	o	o	o	-	o	o	o	-	-
all others	reserved																

Note that “o” means “optional” for the GUI element and “-” means “not used” for the GUI element.

10.3 Link information

There could be several links which may be included in the GUI element data structures.

The *info_type* in Table 10.1 and Table 10.2 describes link information using one byte.

Each link has the basic link information structure, i.e. the text link, bitmap link or sound link structure.

10.3.1 Basic link information structure

The link information is used for reference to the other GUI element or data entity. The structure of link information is specified in this section and is in four bytes. The info_type field in the link information structure is defined in Table 10.1 and Table 10.2.

10.3.1.1 Text link

The text data is defined in section 8.2.1 is used for text, label, hotlink, etc. using text link structure.

The text link structure is defined as follows:

byte count	contents
00 00_{16}	text_type
00 01_{16}	
00 02_{16}	element_ID
00 03_{16}	

Figure 10.1 – Text link data structure

The *text_type* field specifies the *element_type* for text data.

byte count	msb							lsb
00_{16}	1	1	reserved	reserved	text, unicode (0_{16})			
01_{16}	info_type							

Figure 10.2 – Element type for text data entity

The *info_type* field describes a property for text data entity. In case of mandatory information for the text data entity, the info_type value is defined in Table 10.1. In case of optional information for the text data entity, the info_type value is defined in Table 10.2.

The *element_ID* field contains the value which uniquely identifies the GUI element and data entity within the panel subunit.

10.3.1.2 Bitmap link

The image data is defined in section 8.2.1 is used for button, icon, etc. using bitmap link structure.

The bitmap link structure is defined as follows:

byte count	contents
00 00_{16}	image_type
00 01_{16}	
00 02_{16}	element_ID
00 03_{16}	

Figure 10.3 – Bitmap link data structure

The *bitmap_type* field specifies the element_type for image data.

byte count	msb							lsb
00_{16}	1	1	reserved	reserved	data_format			
01_{16}	info_type							

Figure 10.4 – Element type for bitmap data entity

The *data_format* field describes the format of the data entity, and the value of *data_format* is specified in Table 8.3.

The *info_type* field describes a property for bitmap data entity. In case of mandatory information for the bitmap data entity, the info_type value is defined in Table 10.1. In case of optional information for the bitmap data entity, the info_type value is defined in Table 10.2.

The *element_ID* field contains the value which uniquely identifies the GUI element and data entity within the panel subunit.

10.3.1.3 Sound link

The sound link structure is defined as follows:

byte count	contents
00 00_{16}	sound_type
00 01_{16}	
00 02_{16}	element_ID
00 03_{16}	

Figure 10.5 – Sound link data structure

The *sound_type* field specifies the element_type of the sound data.

byte count	msb							lsb
00_{16}	1	1	reserved	reserved	AIFF-C (8_{16})			
01_{16}	info_type							

Figure 10.6 – Element type for sound data entity

The *info_type* field describes a property for sound data entity. In case of mandatory information for the sound data entity, the info_type value is defined in Table 10.1. In case of optional information for the sound data entity, the info_type value is defined in Table 10.2.

The *element_ID* field contains the value which uniquely identifies the GUI element and data entity within the panel subunit.

10.3.1.4 GUI element link

The GUI element link structure is defined as follows:

byte count	contents
00 00_{16}	element_type
00 01_{16}	
00 02_{16}	element_ID
00 03_{16}	

Figure 10.7 – GUI element link data structure

The *element_type* field specifies the element_type of the linked element, i.e. panel, group, GUI element.

byte count	msb							lsb
00_{16}	1	0	reserved					
01_{16}	GUI_element_type							

Figure 10.8 – Element type for the other GUI element

The *GUI_element_type* field describes the meaning and format of the GUI element, and the value of *element_type* is specified in Table 8.1, Table 8.2.

The *element_ID* field contains the value which uniquely identifies the GUI element and data entity within the panel subunit.

10.3.2 Mandatory links

The GUI element has the mandatory links in its data structures, using basic link structure defined in section 10.3.1.

The *info_type* in Table 10.1 describes the type of mandatory link information using one byte.

For detail, these mandatory links is mentioned in each GUI element structure, refer to section 10.5.

10.3.2.1 Label link

Each GUI element has a label, which shows its name or functionality using text, and the *label link* uses a *text link* structure. For details, refer to section 10.3.1.1. When there is no label although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.2 Image link

The image data is used for icon, etc. The image of GUI element is put on the surface of the GUI element. The *image link* uses a *bitmap link* structure. For details, refer to section 10.3.1.2. When there is no image although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.3 Element link

A panel or group can have GUI elements using *element_link* and this link information used for specifies the panel data structure. The *element link* uses a *GUI element link* structure. When there is no element link although it is mandatory, element_ID field shall be set to $FFFF_{16}$. For details, refer to section10.3.1.4. The *info_type* value are defined in Table 10.1

10.3.2.4 Released label link

A BUTTON has a released label, which shows its name or functionality using text in case the BUTTON is not pressed, i.e. the controller will display the released label while the user doesn't interact with it, and the *label link* uses a *text link* structure. For details, refer to section 10.3.1.1. When there is no label although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.5 Released bitmap link

A BUTTON has a released bitmap, which shows its name or functionality using bitmap in case the BUTTON is not pressed, i.e. the controller will display the released bitmap while the user doesn't interact with it, and the *released bitmap link* uses a *bitmap link* structure. For details, refer to section 10.3.1.2. When there is no image although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.6 Pressed label link

A BUTTON has a pressed label, which shows its name or functionality using text while the BUTTON is pressed, and the *label link* uses a *text link* structure. For details, refer to section 10.3.1.1. When there is no label although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.7 Pressed bitmap link

A BUTTON has a pressed bitmap, which shows its name or functionality using bitmap while the BUTTON is pressed, and the *pressed bitmap link* uses a *bitmap link* structure. For details, refer to section 10.3.1.2. When there is no image although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.8 Animation element link

An ANIMATION has plural animation elements, and each of them shows its name or functionality respectively using text and bitmap. The *animation element link* is specified below. When there is no text although in each animation element, element_ID field shall be set to $FFFF_{16}$. When there is no image in each animation element, element_ID field shall be set to $FFFF_{16}$. The *info_type* value is defined in Table 10.1, and both animation element text link and animation element bitmap link in the *animation element link* shall have the same *info_type* value, i.e. the value for animation element link.

The *animation element link* is defined as follows:

byte count	contents
$00\ 00_{16}$	animation_element_text_link
$00\ 01_{16}$	
$00\ 02_{16}$	
$00\ 03_{16}$	
$00\ 04_{16}$	animation_element_bitmap_link
$00\ 05_{16}$	
$00\ 06_{16}$	
$00\ 07_{16}$	

Figure 10.9 – Animation element link structure

The *animation_element_text_link* field is a link to the text data using a *text link* structure. Refer to section 10.3.1.1. This field is called animation element text link.

The *animation_element_bitmap_link* field is a link to the image data using a *bitmap link* structure. For details, refer to section 10.3.1.2. This field is called animation element bitmap link.

10.3.2.9 Choice element link

A CHOICE has plural choice elements, and each of them is a candidate for user choice using text and bitmap. The *choice element link* is specified below. When there is no text although in each choice element, element_ID field shall be set to $FFFF_{16}$. When there is no image in each choice element, element_ID field shall be set to $FFFF_{16}$. The *info_type* values for choice element are defined in Table 10.1. When *info_type* value is 11_{16}, the choice element is not chosen currently. When *info_type* value is 12_{16}, the choice element

is chosen currently. Both *choice element text link* and *choice element bitmap link* in the *choice element link* shall have the same *info_type* value, i.e. the value for choice element link.

The *choice element link* is defined as follows:

byte count	contents
00 00_{16}	choice_element_text_link
00 01_{16}	
00 02_{16}	
00 03_{16}	
00 04_{16}	choice_element_bitmap_link
00 05_{16}	
00 06_{16}	
00 07_{16}	

Figure 10.10 – Choice element link structure

The *choice_element_text_link* field is a link to the text data using a *text link* structure. For details, refer to section 10.3.1.1. This field is called choice element text link.

The *choice_element_bitmap_link* field is a link to the image data using a *bitmap link* structure. For details, refer to section 10.3.1.2. This field is called choice element bitmap link.

10.3.2.10 Text link

A TEXT has a text link, which shows the content using text, and the *text link* uses a *text link* structure. For details, refer to section 10.3.1.1. When there is no text although it is mandatory, element_ID field shall be set to $FFFF_{16}$. The valid GUI elements for this link information and the *info_type* value are defined in Table 10.1

10.3.2.11 Toggle element link

A TOGGLE has plural toggle elements, and each of them is a candidate using text and bitmap, switched to the next one whenever the user selects this GUI element. The *toggle element link* is specified below. When there is no text although in each choice element, element_ID field shall be set to $FFFF_{16}$. When there is no image in each toggle element, element_ID field shall be set to $FFFF_{16}$. The *info_type* values for toggle element are defined in Table 10.1. Both text_link and bitmap_link in the *toggle element link* shall have the same *info_type* value.

The *toggle element link* is defined as follows:

byte count	contents
00 00_{16}	toggle_element_text_link
00 01_{16}	
00 02_{16}	
00 03_{16}	
00 04_{16}	toggle_element_bitmap_link
00 05_{16}	
00 06_{16}	
00 07_{16}	

Figure 10.11 – Toggle element link structure

The *toggle_element_text_link* field is a link to the text data using a *text link* structure. For details, refer to section 10.3.1.1. This field is called toggle element text link.

The *toggle_element_bitmap_link* field is a link to the image data using a *bitmap link* structure. For details, refer to section 10.3.1.2. This field is called toggle element bitmap link.

10.3.3 Optional links

There could be several optional links which may be included in the GUI element data structures.

The *info_type* in Table 10.2 describes link information using one byte.

These optional links will be in an optional link list and the optional link list is put into the GUI element.

Each optional link has the basic link structure, i.e. the text link, bitmap link or sound link structure.

10.3.3.1 Optional link List

Each GUI element data structure contains zero or more optional links, from the items described in 10.2.2. The list of optional links is formatted as follows:

byte count	contents
$00\ 00_{16}$	optional_link[0]
$00\ 01_{16}$	
$00\ 02_{16}$	
$00\ 03_{16}$	
$00\ 04_{16}$	:
:	
:	optional_link[n – 1]
:	
:	
:	

Figure 10.12 – Optional link list structure

The *optional_link[x]* field specifies the reference to the data entities that is a property of the GUI element. The *optional_link[x]* field has either the text link, bitmap link or sound link structure, and the size of each *optional_link[x]* fields is four bytes. In each link structure, and the value of *data_format* is specified in Table 8.3, and the *info_type* field describes optional property for data entity defined in Table 10.2.

The *element_ID* field contains the value which uniquely identifies the GUI element and data entity within the panel subunit.

10.3.3.2 Optional link structures

This section describes the detail of optional link structure, which each GUI element has. The valid optional links for each GUI element is specified in Table 10.2.

10.3.3.2.1 Background pattern link

The pattern data can be used to fill the background of a panel or group. Patterns can be used in place of simple colors, to create more interesting visual presentations, such as textured surfaces which can be similar in nature to the physical device that the panel represents. The *background pattern* link uses a *bitmap link* structure. For details, refer to section 10.3.1.2.

This link can also specify suggestively the background of the element drawn by the controller when the element doesn't receive the focus. In case that the element has this link, the element shall also have the background focused pattern link, then the controller can use the pattern to draw the element when it receives the focus. Therefore the element can have different appearance between when it has the focus and when it dose not. The valid GUI elements for this link and the info_type value are defined in Table 10.2.

10.3.3.2.2 Background picture link

The image data can be used for the background of a panel or group. The *background picture* link uses a *bitmap link* structure. For details, refer to section 10.3.1.2.

This link can also specify suggestively the background of the element drawn by the controller when the element doesn't receive the focus. In case that the element has this link, the element shall also have the background focused picture link, then the controller can use the picture to draw the element when it receives the focus. Therefore the element can have different appearance between when it has the focus and when it dose not. The valid GUI elements for this link and the info_type value are defined in Table 10.2.

10.3.3.2.3 Focus sound link, Select sound link, Pressed sound link, Released sound link

All of these links use a *sound link* structure, For details, refer to section 10.3.1.3.

The sounds are played by the controller when the specified event occurs. In case of focus sound, it is played when a GUI element receives the focus. Select sound is played when the user selects the GUI element. BUTTON could have pressed sound and released sounds instead of select sound. The pressed sound is played when the user pushes the BUTTON element, and the released sound is played when the user release the BUTTON after it is pressed.

All of these links is valid only when a GUI element is interactive.

10.3.3.2.4 Default text link

The default text attribute specifies a default string for text entry using a *text link* structure. For details, refer to section 10.3.1.1. The text data shall be smaller than or equal to maximum text entry data size specified in section 10.4.2.9.

10.3.3.2.5 Hotlink

The hotlink attribute specifies a URL as a text data using a *text link* structure. For details, refer to section 10.3.1.1. The text data contains the URL text, e.g. "http://www.1394TA.org/xxx/yyy".

10.3.3.2.6 Title link

The title can be used for title image of a panel or group. The title link use a *bitmap link* structure, For details, refer to section 10.3.1.2.

When this link exists in a GUI element, then the controller knows that this element represents the title of a panel or a group. In scaled-down display situations, the controller may wish to have this knowledge in order to keep the title displayed for the user at all times (or some other display strategy).

10.3.3.2.7 Max label link, Min label link, Center label link, Unit label link

These label links are text strings and use a *text link* structure. For details, refer to section 10.3.1.1. They are used for SHOW RANGE and SET RANGE in which a minimum, center, maximum or unit label might be useful when rendering the element on the display. Those GUI elements could have only some of those links, e.g. the center value and its unit could be in the center label without a unit label.

10.3.3.2.8 Launch bitmap link

Some interactive GUI element, e.g. ENTRY, CHOICE, could have launch bitmaps. The launch bitmap is applied when the GUI element is displayed, and the user input window appears using controller dependent manner when the GUI element is selected. In this case, no user action is transmitted to the target until the user inputs the data or selects elements. This link uses a *bitmap link* structure. For details, refer to section 10.3.1.2.

10.3.3.2.9 Focused label link

This label link is text string and uses a text link structure. For details, refer to section 10.3.1.1. This label can be used to show its name or functionality using text when the element receives the focus, instead of the label used when it dose not receive the focus.

In case that animation element has this link, the number of focused label links in the element must be equal to the number of animation element links. In case that the toggle element has this link, the number of focused label links in the element must be equal to the number of toggle element links.

The valid GUI elements for this link information and the info_type value are defined in Table 10.2. This may be used with Focused bitmap link.

10.3.3.2.10 Focused bitmap link

The focused bitmap link specifies the appearance using bitmap when the element receives the focus, instead of the bitmap used when it dose not receive the focus. For details of the link structure, refer to section 10.3.1.2.

In case that a BUTTON element has this element, the pressed bitmap may show a image when the BUTTON is focused and pressed. In case that animation element has this link, the number of focused bitmap links in the element must be equal to the number of animation element links. In case that toggle element has this link, the number of focused bitmap links in the element must be equal to the number of toggle element links.

The valid GUI elements for this link information and the info_type value are defined in Table 10.2. This may be used with Focused label link.

10.3.3.2.11 Background focused picture link

The background focused picture link specifies suggestively the background of the element drawn by the controller when the element receives the focus. In case that the element has this link, the element shall also have the background picture link, then the controller can use the picture to draw the element when it dose not receive the focus. Therefore the element can have different appearance between when it has the focus and when it dose not.

For details of the link structure, refer to section 10.3.1.2. The valid GUI elements for this link information and the info_type value are defined in Table 10.2. The element that has this link shall have neither background focused pattern link nor background focused color attribute.

10.3.3.2.12 Background focused pattern link

The background focused pattern link specifies suggestively the background of the element drawn by the controller when the element receives the focus. In case that the element has this link, the element shall also have the background pattern link, then the controller can use the pattern to draw the element when it dose not receive the focus. Therefore the element can have different appearance between when it has the focus and when it dose not.

For details of the link structure, refer to section 10.3.1.2. The valid GUI elements for this link information and the info_type value are defined in Table 10.2. The element that has this link shall have neither background focused picture link nor background focused color attribute.

10.4 Attributes

The GUI element has mandatory and optional attributes in its structure.

The *info_type* in Table 10.1 and Table 10.2 describes attributes using one byte.

Each attribute has the same structure specified in basic attribute structure.

10.4.1 Basic attribute structure

The attribute is used for specifying a property of GUI element. The basic structure is defined in this section.

The basic format for attribute is as follows:

byte count	contents
00 00_{16}	info_type
00 01_{16}	attribute_length
:	type_specific_attribute
:	

Figure 10.13 – Basic attribute structure

The info_type field specifies one of the info_type values defined in Table 10.1 and Table 10.2.

The *attribute_length* field specifies the size of the type_specific_attribute field using one byte.

The *type_specific_attribute* field specifies the property of each attribute. The structure of each attribute is defined in the following sections.

10.4.2 Mandatory Attributes

The GUI element has the mandatory attributes in its data structures, using basic attribute structure defined in section 10.4.1. The *info_type* in Table 10.1 describes the type of mandatory attribute using one byte.

For detail, The *type_specific_attribute* field structure for these mandatory attributes are defined in the following section, and these mandatory attributes have basic attribute structure with *type_specific_attribute* field in each GUI element described in section10.5.

10.4.2.1 Width height

Most GUI elements define a width_height attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	width
01_{16}	
02_{16}	height
03_{16}	

Figure 10.14 – Width height attribute structure

The *width* and *height* fields specify the size, in pixel, of GUI element. At least, one of bitmaps in the GUI element shall be smaller than the size specified by *width* and *height*.

10.4.2.2 Interactive

Most GUI elements define an interactive attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	activity							

Figure 10.15 – Interactive attribute structure

The *activity* field specifies whether the GUI element is interactive, temporary non-interactive or display only. When active bit is set to zero, then the interactivity of the GUI element is enable, i.e. the user can interact with the GUI element and the controller sends a user action to the target, e.g. click it.

If *activity* field value is set to one, then the interactivity of the GUI element is disable temporary, i.e. the user cannot interact with the GUI element, and the controller shall not send a user action to the target but the GUI element is still visible, e.g. the controller make it gray out and doesn't move the focus to it.

When *activity* field value is set to two, then the interactivity of the GUI element is display only, i.e. the user cannot interact with the GUI element and the controller shall not send a user action to the target and the

GUI element is displayed clearly, e.g. just information to the user so the controller doesn't move the focus to it.

In case of panel link, if the *activity* field value is set to two, then the user can interact with the GUI element but the controller shall not send a user action to the target and the panel link is displayed clearly. The controller will pull the panel data indicated by the panel link according to the user action.

The other values for *activity* field are reserved for future extension.

10.4.2.3 Aspect ratio

A PANEL defines an aspect_ratio attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	aspect_ratio							

Figure 10.16 – Aspect ratio attribute structure

The *aspect_ratio* field value is defined in the table below:

Table 10.3 – Aspect_ratio field value

aspect_ratio	ratio	note
00_{16}	1.0	square pixel
01_{16}	1.067	720 by 576 pixels rendered on 4 by 3 display
02_{16}	0.909	704 by 480 pixels rendered on 4 by 3 display
03_{16}	0.889	720 by 480 pixels rendered on 4 by 3 display
04_{16}	1.422	720 by 576 pixels rendered on 16 by 9 display
05_{16}	1.212	704 by 480 pixels rendered on 16 by 9 display
06_{16}	1.185	720 by 480 pixels rendered on 16 by 9 display
all other values	all other values	

Note that the size of display written in above table is neither specifies the controller's screen size nor panel size. It shows only aspect ratio of the supposed by the panel subunit. The controller may not have the capability specified by aspect ratio attribute, and the safety display area size of a panel is 640 x 480, i.e. the controller may display non-square pixel images on its screen with square pixel.

Except for 640 x 480 safety display area size, the aspect_ratio field value is set to zero, i.e. square pixel.

10.4.2.4 Linked panel ID

A PANEL LINK defines a linked_panel_ID attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	panel_identifier
01_{16}	
02_{16}	
03_{16}	

Figure 10.17 – Linked panel ID attribute structure

The *panel_identifier* field contains the element_type and the element_ID of a PANEL , HELP PANEL or ALERT PANEL using element identifier structure specified in section 7.1, which should be shown to the user, if the PANEL LINK GUI element is selected. When the *panel_identifier* field value is $FFFF_{16}$, then the controller should display the latest displayed PANEL except for HELP PANEL and ALERT PANEL.

10.4.2.5 Animation type

An ANIMATION defines an animation type attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	repetition							
01_{16}	speed							

Figure 10.18 – Animation type attribute structure

The *repetition* field specifies how to play the images. The controller only once displays the sequence of the animation elements when *repetition* value is zero, and repeatedly displays them when *repetition* value is one. In case *repetition* value is two, the controller would play the animation elements backwards until it reaches the beginning when it reaches the end, and then start playing forwards again. The other values for *repetition* field are reserved for future extension.

The *speed* field specifies how fast the animation element should be switched to the next animation element on the display screen. This value is eight bits numeric number in units of 0.1 second. However this value is just suggestion to the controller, so animation element could be changed at the controller dependent timing, e.g. refresh of the display. If the *speed* field value is FF_{16}, then the switching speed to the next animation element is controller dependent.

The other values for *speed* field are reserved for future extension.

10.4.2.6 Range type

A Range, i.e. SHOW RANGE and SET RANGE, defines a range type attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	range_shape_type							
01_{16}	value_range							
02_{16}								
03_{16}	step_value							
04_{16}								

Figure 10.19 – Range type attribute structure

The *range_shape_type* field specifies the shape of range. When *range_shape_type* field is set to zero, this range is horizontal range, and when this field is set to one, it is vertical range, and when this field is set to two, it is dial type range. The other values for *range_shape_type* field are reserved for future extension.

It is allowed for the controller to ignore the *range_shape_type* if it doesn't support the specified type.

The *value_range* field specifies the unsigned value range value of the range element, which is sixteen bits numeric number. This value is relative value, the panel subunit could specify the other value to the user by using the minimum and maximum label or value offset optional attribute.

The *step_value* field specifies the unsigned step value of this range, which is sixteen bits numeric number.

10.4.2.7 Value set

Range defines a value set attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	value_set							
01_{16}								

Figure 10.20 – Value set attribute structure

The *value_set* field specifies the current value of this range, which is sixteen bits unsigned integer value.

10.4.2.8 Entry type

An ENTRY defines an entry type attribute; this is defined as follows:

byte count	msb							lsb
00_{16}	input_data_type							
01_{16}	entry_qualifier							

Figure 10.21 – Entry type attribute structure

The *input_data_type* field specifies the type of input data, i.e. textual data, numeric number or date information. The *input_data_type* field value is specified in Table 10.4.

The *entry_qualifier* field specifies whether the input characters should be displayed in the entry field on the screen or not. When *entry_qualifier* field is set to zero, the input characters are concealed, and when this field is set to one, the input characters are visible. The other values are reserved for future extension.

The following table defines the input data type:

Table 10.4 – Input data type field value

input_data_type	meaning	comment
0_{16}	text entry	This ENTRY is used for text input. After finishing input of the text string, the controller shall send the text string to the panel subunit using USER ACTION command.
1_{16}	numeric entry	This ENTRY is used for natural numeric number input. After finishing input of the number, the controller shall send the numeric number to the panel subunit using USER ACTION command in thirty-two bits binary value.
2_{16}	float entry	This ENTRY is used for a real number input. After finishing input of the number, the controller shall send the float number to the panel subunit using USER ACTION command in thirty-two bits floating point value.
3_{16}	date time	This ENTRY is used for date time input. After finishing input of the date time information, the controller shall send the date time information, whose format is specified in section 10.5.8, to the panel subunit using USER ACTION command.
all other values		Reserved for future specification

10.4.2.9 Entry type depend

An ENTRY defines an entry type depend attribute; this is defined as follows:

byte count	msb							lsb
00_{16}	entry_type_depend_info							
01_{16}								
02_{16}								
03_{16}								
04_{16}								
05_{16}								
06_{16}								
07_{16}								

Figure 10.22 – Entry type depend attribute structure

The *entry_type_depend_info* field specifies maximum input data size or default entry value.

In case of a text entry, the *entry_type_depend_info* field contains maximum input data size in byte using text entry string size data structure specified in section 10.5.8. The default value for text entry is used for text data entity.

In case of numeric and float, the *entry_type_dependent_info* field contains four bytes value using numeric value data structure specified in section 10.5.8.

10.4.2.10 Choice type

A CHOICE defines a choice type attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	choice_number_type							
01_{16}	choice_number							

Figure 10.23 – Choice type attribute structure

The *choice_number_type* field specifies the number of choice elements with *choice_number* field. When *choice_number_type* field is set to zero, this means “EQUAL”, and when this field is set to one, this means “MORE_THAN”, and when this field is set to two, this means “LESS_THAN”.

The other values for *choice_number_type* field are reserved for future extension.

The *choice_number* field specifies the base number of choice elements and this field contains eight bits numeric number. If this element request only one choice, then *choice_number_ type* is set to be zero and *choice_number* is one. The controller should check the input selection according to choice type and choice number.

10.4.2.11 Status type

A STATUS defines a status type attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	status_type							

Figure 10.24 – Status type attribute structure

The *status_type* field specifies the type of status element, i.e. hourglass or barber-pole. When *status_ type* field is set to zero, it is suggested this status element is hourglass, and when this field is set to one, it is suggested this status element is barber-pole. The other values for *status_type* field are reserved for future extension. It is allowed for the controller to use the other look and feel, if it doesn't support the specified type.

10.4.2.12 Status working

A STATUS defines a status working attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	status_working							

Figure 10.25 – Status working attribute structure

The *status_working* field specifies whether this is in motion or static. When *status_working* field is set to zero, this element is static, and when this field is set to one, this element is working. The other values for *status_working* field are reserved for future extension.

10.4.2.13 Toggle status

A TOGGLE defines a toggle status attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	toggle_status							

Figure 10.26 – Toggle status attribute structure

The *toggle_status* specifies the current status of the toggle. The *toggle_status* field value specifies an index number of the toggle elements and the index number specifies what number the current element is in the line of toggle elements, i.e. the index number shows natural number from one to the number of elements.

10.4.2.14 Content availability

A CONTENT ICON defines a content availability attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	reserved					schedule	working	aoc

Figure 10.27 – Content availability attribute structure

The *schedule* bit specifies whether the target will output the content at the specified time by *broadcast date* attribute. When the *schedule* bit is set to one, the target will output the stream data at the specified time by *broadcast date* attribute. The *schedule* bit shall be set to zero and there is no *broadcast date* attribute in the CONTENT ICON structure, if the target inputs the stream data.

The *working* bit specifies whether the target is now receiving/sending the stream data or not. When set to zero, the target is not working now. When set to one, the target is now inputting from the specified plug or outputting the content indicated by CONTENT ICON on the specified plug in the AV *plug* attribute.

The *aoc* bit is the *abailability_of_content* bit. When set to zero, the indicated content is not available now or the target is not ready to input the stream data as the indicated content or action by the CONTENT ICON

according to user action. When set to one, the indicated content is now available or the target can input the stream data as the indicated content or action.

10.4.2.15 AV Plug

The AV *plug* attribute is used to specify an isochronous channel of AV stream which is associated with the GUI element. When the GUI element is interacted by the user, the stream data will be on the plug specified by AV plug attribute.

The *type_specific_attribute* field of *AV_plug* attribute is defined as follows:

byte count	msb							lsb
00_{16}	plug							
01_{16}	reserved	in_out	fmt					
02_{16}	(msb)							
03_{16}	fdf							
04_{16}								(lsb)
05_{16}	stream_specific_data							
06_{16}								
07_{16}								

Figure 10.28 – AV plug attribute structure

The *plug* field specifies the unit plug that is carrying the audio and/or video data. The value of plug field is specified as follows;

Table 10.5 – Plug field value

plug field value	meaning
00_{16} – $1E_{16}$	Serial Bus oPCR[0] – oPCR[30] or iPCR[0] – iPCR[30]
all other values	Reserved for future extension

The *in_out* bit specifies the target behavior when this GUI element is interacted by the user, i.e. input or output stream data. When set to zero, the target will input the stream data from the specified plug by *plug* field. When set to one, the target will output the stream data to the specified plug by *plug* field.

The *fmt* and *fdf* fields are defined in IEC 61883 [R6], Digital Interface for Consumer Electronic Audio/Video Equipment. Together they specify the signal format for the Serial Bus output plug.

The *stream_specific_data* field specifies the stream dependent information that identified by *fmt* field.

In case of DV stream, *fmt* field shall be set to $00\ 0000_2$ and *stream_specific_data* field shall be filled with zero.

In case of MPEG 2 transport stream, *fmt* field shall be set to $10\ 0000_2$ and *stream_specific_data* field is as follows,

byte count	msb							lsb
05_{16}	PMT_PID							
06_{16}								
07_{16}	reserved							

Figure 10.29 – Stream specific data field structure for MPEG 2 TS

The *PMT_PID* field specifies the PID of the Program Map Table (PMT) packet, specified in reference [R13]. It is controller dependent which stream specified in the PMT packet is played back.

The other case, *stream_specific_data* field shall be filled with zero.

10.4.3 Optional Attributes

There are several optional attributes which may be included in the GUI element data structures.

The *info_type* in Table 10.2 describes optional attributes using one byte.

These optional attributes will be in an optional attribute list and the optional attribute list is put into the GUI element, refer to section 10.5.

10.4.3.1 Optional Attributes List

Each GUI element data structure contains zero or more optional attributes from the items described in section 10.2.2 by using optional attributes list. This list of optional attributes (not a descriptor list; just an array of data structures) is formatted as follows:

byte count	contents
$00\ 01_{16}$	optional_panel_attribute[0]
:	
:	
:	:
:	optional_panel_attribute[n – 1]
:	
:	

Figure 10.30 – Optional attribute list structure

The optional_panel_attribute[x] fields specify one of the optional attributes, each optional_panel_attribute[x] field is the basic attribute format defined in 10.4.1.

10.4.3.2 Optional attribute structure

This section describes the detail of optional attribute structure, which each GUI element has. The valid optional attributes for each GUI element are specified in Table 10.2.

For detail, The *type_specific_attribute* field structure for optional attributes are defined in the following section, and these optional attributes have basic attribute structure with *type_specific_attribute* field in each GUI element described in section 10.5.

10.4.3.2.1 Position

Most GUI elements can define an optional position attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	x_position
01_{16}	
02_{16}	y_position
03_{16}	

Figure 10.31 – Position attribute structure

The *x_position* and *y_position* fields specify the position of the upper left corner of the GUI element, relative to its container. If the GUI element is in a panel, these values are relative to the upper left corner of the panel. If the GUI element is in a group, these values are relative to the upper left corner of the group. In case of panel, these values are relative to the upper left corner of the safety display area. The x_position value is positive incremented from left to right, and the y_position value is positive incremented from top to bottom. At the upper left corner, x_position is zero and y_position is zero.

If a GUI element in a group has a position attribute, all GUI elements in the group shall have position attribute. If a directly linked GUI element from a panel has position attribute, all directly linked GUI elements from the panel shall have a position attribute.

10.4.3.2.2 Font size

Most GUI elements can define an optional font size attribute. This attribute is used to specify the font size of labels and text data in GUI element and it is an unsigned value encoded; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	font_size

Figure 10.32 – Font size attribute structure

The font_size field is defined for the target to assist the controller with construction of the GUI display using one byte. The panel subunit does not specify any particular font face or other attributes that would typically be found in a text editing application on a PC. Rather, basic text is defined using the attribute font size, as defined in the table below:

Table 10.6 – Font_size field value

font_size	meaning	suggested Pixel size
00_{16}	small	12 x 24 pixels
01_{16}	medium	16 x 32 pixels
02_{16}	large	20 x 40 pixels
all other values	reserved for future specification	

The selection of what constitutes a “small, medium, or large” font is up to the controller. Typically, a controller will ship with one standard “text display” font in three sizes. Each GUI element which needs to display text will be displayed using this font in the size specified by the GUI element data structure.

If a panel subunit wants to make sure that text is displayed with a specific set of characteristics such as typeface, font size, color, etc., then it should use a bitmap graphic icon of the text for that GUI element, instead of a standard text GUI element.

10.4.3.2.3 Relation

The *relation* attribute is used to specify an arbitrary set of GUI elements which the controller should try to display together, in situations where it must present a partial display. All GUI elements which share the same relation attribute value should be displayed together, if possible.

The *relation* attribute is an unsigned value encoded; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	relation
01_{16}	

Figure 10.33 – Relation attribute structure

10.4.3.2.4 Focus navigation

The *focus_navigation* attribute specifies where the controller should go next, when moving the focus between GUI elements. A GUI element which has this attribute points to some other GUI element, which should be the one to next receive the focus. This attribute is valid only when the GUI element is interactive. The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	up_GUI_element_ID
01_{16}	
02_{16}	down_GUI_element_ID
03_{16}	
04_{16}	left_GUI_element_ID
05_{16}	
06_{16}	right_GUI_element_ID
07_{16}	

Figure 10.34 – Focus navigation attribute structure

The up/down/left/right_GUI_element_ID fields each specify the element_ID of the next GUI element in that “direction” of navigation. Only the four basic directions are specified in this mechanism. When the user navigates with the input mechanism of the controller, the controller decides which direction is being selected and then navigates to the specified GUI element.

If the controller has adjusted the display, then it is free to alter the direction of navigation in a manner which makes sense based on the new layout of the GUI elements.

10.4.3.2.5 Help panel ID

Most GUI elements can define an optional help panel ID attribute. The help panel ID attribute points to a HELP PANEL which contains help information for the specified GUI element. The controller is free to supply some means for providing access to triggering the display of help information for GUI elements; when this mechanism is accessed by the user, the help_panel_link attribute points to the help information for that element. The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	help_panel_identifier
01_{16}	

Figure 10.35 – Help panel ID attribute structure

The help_panel_identifier field contains the element_ID of the HELP PANEL.

Note that the controller and the target shall not change the current panel when the controller pulls the panel data of HELP PANEL because help panel is limited to temporary usage. The HELP PANEL doesn't contain interactive GUI elements, but HELP PANEL can be linked to the other PANEL or HELP PANEL using a PANEL LINK element.

10.4.3.2.6 Background color

A PANEL can have an optional background color attribute, is encoded as a *color data* structure. The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
0000_{16}	alpha
0001_{16}	red
0002_{16}	green
0003_{16}	blue

Figure 10.36 – Color data structure

The *alpha* field specifies the transparency of the pixel with respect to the background. A value of zero indicates that the pixel is entirely transparent, i.e. back ground is fully visible, and a value of FF_{16} indicates that the pixel is opaque, i.e. back ground is not visible at all. The other value represents intermediate transparencies approximately as fractional opacity, alpha/255.

The red, green and blue fields shall specify the intensity of their respective color, where zero is off and FF_{16} represent full intensity. The other value presents intermediate intensities approximately as a faction of 255.

This attribute can also specify suggestively the background of the element drawn by the controller when the element doesn't receive the focus. In case that the element has this attribute, the element shall also have the focused background color attribute, then the controller can use the color to draw the element when it receives the focus. Therefore the element can have different appearance between when it has the focus and when it dose not. The valid GUI elements for this attribute and the info_type value are defined in Table 10.2.

10.4.3.2.7 Audio video

The audio video attribute is used to specify an isochronous channel of AV data which is associated with the GUI element. When the GUI element is being displayed to the user, the data associated with the audio video attribute will also be displayed.

The audio video attribute has highest priority to the other related attributes. In case a panel or a group has audio video attribute and the controller can display the video data specified by audio video attribute, then the controller should display the back ground video specified by either back ground color, back ground picture, back ground pattern or audio video attribute. The presentation priority is,

audio video > back ground picture > back ground pattern > back ground color.

In case a panel or a group has *audio_video* attribute and the controller can play the audio data specified by audio video attribute, then the controller should play the music specified by audio video attribute in stead of focus sound when the panel or group is focused.

In case an ICON or a CONTENT ICON has audio video attribute and the controller can display the video data specified by audio video attribute, then the controller should display the video instead of the icon_bitmap or content_icon_bitmap. In case an ICON or a CONTENT ICON has audio video attribute and the controller can play the specified audio data, then the controller should play the sound specified in audio video attribute instead of focus sound attribute.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	plug							
01_{16}	2		fmt					
02_{16}	(msb)							
03_{16}	fdf							
04_{16}								(lsb)
05_{16}	stream_specific_data							
06_{16}								
07_{16}								

Figure 10.37 –Audio video attribute structure

The *plug* field specifies the unit plug that is carrying the audio and/or video data. The value of plug field is specified as follows;

Table 10.7 – Plug field value

plug field value	meaning
00_{16} – $1E_{16}$	Serial Bus oPCR[0] – oPCR[30] or iPCR[0] – iPCR[30]
all other values	Reserved for future extension

The *fmt* and *fdf* fields are defined in IEC61883[R6], Digital Interface for Consumer Electronic Audio/Video Equipment. Together they specify the signal format for the Serial Bus output plug.

The *stream_specific_data* field specifies the stream dependent information that identified by *fmt* field.

In case of DV stream, *fmt* field shall be set to $00\ 0000_2$ and *stream_specific_data* field shall be filled with zero.

In case of MPEG 2 transport stream, *fmt* field shall be set to $10\ 0000_2$ and *stream_specific_data* field is as follows,

byte count	msb							lsb
05_{16}	PMT_PID							
06_{16}								
07_{16}	reserved							

Figure 10.38 – Stream specific data field structure for MPEG 2 TS

The *PMT_PID* field specifies the PID of the Program Map Table (PMT) packet, specified in reference [R13]. It is controller dependent which stream specified in the PMT packet is played back.

The other case, *stream_specific_data* field shall be filled with zero.

10.4.3.2.8 Show with parent

The *show with parent* attribute indicates whether the specified GUI element should be displayed along with its parent (containing) element. Currently, this attribute is defined only for panels; when the attribute is present and set to true, the controller should try to display the panel with its "parent" panel. The definition of a parent panel is the one which comes before this one in the hierarchy. The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	reserved							swp

Figure 10.39 – Show with parent attribute structure

The *swp* bit is the *show_with_parent* bit. When set to one, then this attribute is "true", and the controller should attempt to show with parent as described above.

10.4.3.2.9 Initial focus

A PANEL can have an optional initial focus attribute. The initial focus attribute specifies which GUI element will be focused when the PANEL is active on the screen for the first time.

A GROUP can also have an optional initial focus attribute to specify the initially focused element when the cursor moves into the GROUP. When only one group is displayed, this attribute specifies which GUI element inside the group will be focused for the first time. If the PANEL has a GROUP and the GUI element in the GROUP is intended to be focused when the PANEL is active for the first time, then the PANEL's initial focus attribute should specify the GROUP and the GROUP's initial focus attribute should specify the GUI element to be focused. In case the PANEL has plural GROUPs, each GROUP may specify its own initial focus.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	focussed_element_type
01_{16}	
02_{16}	focussed_element_ID
03_{16}	

Figure 10.40 – Initial focus attribute structure

The focussed_element_type field contains the element_type of the focussed GUI element in the panel.

The focussed_element_ID field contains the element_ID of the focussed GUI element in the panel.

10.4.3.2.10 Value offset

A Range, i.e. SHOW RANGE and SET RANGE can have an optional value offset attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	value_offset							
01_{16}								

Figure 10.41 – Value offset attribute structure

The *value_offset* field specifies the signed offset value of the range element, i.e. SHOW RANGE and SET RANGE, which is sixteen bits numeric number. With value power attribute and this attribute, the controller can calculate real current value for the range element without interaction with target. If the range element don't have both the value offset attribute and the value power attribute, the controller shall not display the real current value at controller side.

The real current value at the controller side is calculated by the following formula;

(real current value) = {(current value) + (value offset)} X 10 $^{(value\_power)}$.

Note that current value is counted in unit of step value and is used only inside the controller, and the current value is reported by value set attribute when the target specifies it.

10.4.3.2.11 Value power

Range can have an optional value power attribute with value offset attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	value_power							
01_{16}								

Figure 10.42 – Value power attribute structure

The *value_power* field specifies the signed power value of the range element, i.e. SHOW RANGE and SET RANGE, which is sixteen bits numeric number.

10.4.3.2.12 Play back time

The play back time attribute shows the required duration of play back the content (program). All values of the play back time attribute are encoded as BCD format such as XXXX hour, XX minutes, XX seconds, here each X shows the four bits BCD value.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	hour
01_{16}	
02_{16}	minuets
03_{16}	second

Figure 10.43 – Play back attribute structure

The *hour*, *minuets*, *second* fields show the play back time. If the target doesn't want to notify the user of some of these field values, these fields shall be filled up with ones.

10.4.3.2.13 Recorded date

The recorded date attribute indicates the date recorded in a storage device, and it should be in past. All values of the recorded date attribute are encoded as BCD format such as XXXX year, XX month, XX date, XX hour, XX minutes, XX seconds, here each X shows the four bits BCD value.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	year
01_{16}	
02_{16}	month
03_{16}	date
04_{16}	hour
05_{16}	minuets
06_{16}	second
07_{16}	reserved

Figure 10.44 – Recorded date attribute structure

The year, month, date, hour, minuets, second fields show the recorded date. If the target doesn't want to notify the user of some of these field values, these fields shall be filled up with ones.

10.4.3.2.14 Broadcast date

The broadcast date attribute indicates the date outputting the specified content by a broadcast receiving device such as tuner or the device which executes a scheduled action. Normally it indicates future date time, but it may be a past date time. If the broadcast date is in past, then it means the content is in broadcasting or playing back in the target device, and it is indicated by content availability attribute whether the content is outputting from the target. All values of the recorded date attribute are encoded as BCD format such as XXXX year, XX month, XX date, XX hour, XX minutes, XX seconds, here each X shows the four bits BCD value.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	year
01_{16}	
02_{16}	month
03_{16}	date
04_{16}	hour
05_{16}	minuets
06_{16}	second
07_{16}	reserved

Figure 10.45 – Broadcast date attribute structure

The year, month, date, hour, minuets, second fields show the recorded date. If the target doesn't want to notify the user of some of these field values, these fields shall be filled up with ones.

10.4.3.2.15 Vendor dependent info

The *vendor dependent info* attribute shows arbitrary information of the vendor. The data length is defined by the owner of this information in the type_specific_attribute_length field.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	(msb)
01_{16}	company_ID
02_{16}	(lsb)
03_{16}	vendor_specific_info
:	

Figure 10.46 – Vendor dependent info attribute structure

The *company_ID* field shall contain the 24-bit unique ID obtained from IEEE Registration Authority Committee.

The *vendor_specific_info* field contains the vendor-specified information and identified by the *company_ID* field. The size of vendor dependent info attribute is specified in *attribute_length* field in basic attribute structure.

10.4.3.2.16 Safety area position

The safety area position attribute shows the position of safety display area on the controller's screen.

In case the controller's screen is larger than the size of safety display area then the controller should display the panel following to this attribute.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	reserved		right	center	left	top	mid	bottom

Figure 10.47 – Safety area position attribute structure

The *right* bit shows the position of the right end on the screen.

The *center* bit shows the position of the horizontal center on the screen.

The *left* bit shows the position of the left end on the screen.

The *top* bit shows the position of the top on the screen.

The *mid* bit shows the position of the vertical center on the screen.

The *bottom* bit shows the position of the bottom on the screen.

When left and bottom bit are set to one, controller locates the panel at the left bottom corner.

When only top bit is set to one, controller locates the panel at the top of screen and horizontal position is controller dependent.

At most one bit could be set to one among right, center, left bits, and zero or one bit could be set to one among top, mid and bottom bits. Furthermore, at most two bits could be set to one among bits in this attribute.

All attributes are just suggestion for controller, so the controller may display GUI elements without enough consideration of the attributes.

10.4.3.2.17 Overlay

The *overlay* attribute is the display indication initiated by the panel subunit. Anytime, the panel subunit can send the overlay GUI element to request the controller to display it on the screen in order to alert the user, etc. In this case, the current panel of the panel subunit must not be changed, and device generation number may not be changed. The overlay GUI element may or may not interactive. The overlay GUI element shall be or include an interactive icon if it is interactive, and it can consist of plural GUI elements using panel data hierarchy.

The panel subunit shall send the overlay GUI element in the one frame.

The controller will make the final decision whether or not it displays this overlay GUI element.

The GUI element in the top hierarchy shall have overlay attribute and the GUI element under the top GUI element may not have overlay attribute when the overlay GUI element has hierarchy structure like panel.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	overlay	duration						

Figure 10.48 – Overlay attribute structure

The *overlay* bit is the *overlay* bit. When set to one, then the attached data is the overlaying GUI element, and the controller shall try to display the overlaying GUI element in front of the screen. When set to zero, then the controller shall try to remove the overlaid GUI element on the screen. When panel subunit stops overlaying, it may send only the top GUI element in the overlay GUI element hierarchy.

The *duration* field shows the duration in which the controller displays the overlaying GUI element in unit of 0.1 second. When set to zero, then the overlaying GUI element is displayed until controller receives the GUI element whose overlay bit is zero.

10.4.3.2.18 Action data size

The action data size attribute shows the size of the user action data in a command frame. This attribute is valid only for the choose_list for the CHOICE element and the enter_data for ENTRY element.

If the element has action data size attribute, the size of the action_data in a user action command shall be smaller than or equal to the size shown by action data size.

When the panel subunit can receive any action_data whose size is limited by the other attributes, e.g. entry data size, this attribute doesn't exist in the GUI element.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	reserved							(msb)
01_{16}	data_size							(lsb)

Figure 10.49 – Action data size attribute structure

The *data_size* field specifies the data size of the action_data in a command frame.

10.4.3.2.19 AV plug info

The *AV plug info* attribute is used to specify the unit plug number whose state the target may change after it accepts the user action for the GUI element, e.g. start outputting, change stream data type. The *AV plug info* attribute is the same structure as *AV plug* attribute specified in section 10.4.2.15.

For instance, if the BUTTON has the *AV plug info* attribute then it means the data will be output from/ input to the specified plug or changed when it is pressed or released

In case the ENTRY has the *AV plug info* attribute and it request to set date information then it means a scheduled action, e.g. timer recording from the specified plug, indicated by its label or bitmap.

10.4.3.2.20 Cursor through

An ICON can have an optional cursor through attribute; the *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	cursor_through							
01_{16}	exit_element_identifier							
02_{16}								
03_{16}								
04_{16}								

Figure 10.50 – Cursor through attribute structure

The *cursor_through* field specifies whether or not the controller sends the PASS THROUGH control command of cursor navigation, i.e. up, down, left, right, right-up, right-down, left-up, left-down and select operation to the panel subunit according to user operation on the ICON. The controller considers the cursor through attribute as disable, if this attribute doesn't exist. When *cursor_through* field is set to zero, the cursor through attribute is disable, i.e. the controller shall not send PASS THROUGH command of navigation and select operation to the panel subunit. In this case the controller manages the cursor movement and sends USER ACTION control command, e.g. select action, according to the user operation.

When this field is set to one, the cursor through attribute is enable, i.e. the controller should send PASS THROUGH command with cursor navigation and select operation to the panel subunit according to user operation. During cursor through is valid, the controller should not send USER ACTION command for this ICON. The other values for *cursor_through* field are reserved for future extension.

The *exit_element_identifier* field specifies the destination GUI element of the cursor when it moves out from this ICON. The *exit_element_identifier* field is valid only when the target changes *cursor_through* field value from enable to disable. When the ICON's *cursor_through* field changed from enable to disable, the first focus position shall be set to the specified GUI element that the target wants.

If the field is filled up with ones, the controller shall choose arbitrary GUI element as the cursor destination.

For example, after cursor comes to the ICON that has a cursor through attribute, PASS THROUGH command transfers a user operation for cursor movement. The panel subunit can updates the surface bitmap according to the user operation, so the cursor might appear in the ICON and could move inside the ICON. When the panel subunit finish interacting by PASS THROUGH command, it changes the cursor through attribute to disable. After that, the controller manages cursor movement, so the cursor moves out from the ICON to the GUI element that *exit_element_identifier* field specifies.

10.4.3.2.21 Assured area size

The *assured area size* attribute requests the controller to display the GUI element completely as specified by width height attribute. So the controller shall not display the GUI element if it cannot assign enough space to render the GUI element, i.e. the controller shall not render even label for the GUI element. Group and ICON can have an emergency element for the time when the GROUP or ICON cannot be displayed because the area size for the GUI element is too small.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	msb							lsb
00_{16}	emergency_element_link							
01_{16}								
02_{16}								
03_{16}								

Figure 10.51 – Assured area size attribute structure

The *emergency_element_link* field specifies the alternative element_link for the case that GROUP or ICON cannot be displayed. The element specified by *emergency_element_link* must be same element type as the original GUI element.

When a panel controller could not display the element for some reason, the controller uses the linked alternative small element to render on the controller's screen, or uses linked error messages to announce to the user.

10.4.3.2.22 Background focused color

The background focused color specifies suggestively the background of the element drawn by the controller when the element receives the focus. In case that the element has this attribute, the element shall also have the background color attribute, then the controller can use the color to draw the element when it dose not receive the focus. Therefore the element can have different appearance between when it has the focus and when it dose not.

For details of color data structure, refer to section 10.4.3.2.6. The valid GUI elements for this attribute and the info_type value are defined in Table 10.2. The element that has this attribute shall have neither background focused picture link nor background focused pattern link.

10.4.3.2.23 Data entity size

The data entity size attribute specifies the size of the data entity in bytes. All GUI elements except STATUS can have one or more this attributes according to the number of data entities in the GUI element.

The *type_specific_attribute* field of this attribute is defined as follows:

byte count	contents
00_{16}	element_identifier
01_{16}	
02_{16}	
03_{16}	
04_{16}	data_size
05_{16}	
06_{16}	
07_{16}	

Figure 10.52 – Data entity size attribute structure

The *element_identifier* field specifies the data entity that contains text, audio or bitmap data. Note that element identifier in the data entity size attribute shall not identify GUI element and the other data entity than directly linked one from the GUI element that has this attribute.

The GUI element that has large bitmap or audio data, e.g. full screen bitmap or over 50k-byte data, should have this attribute for such big data. The GUI element should not have this attribute for label text data.

The *data_size* field specifies the size of content data in bytes, i.e. unsigned thirty-two bit integer value.

10.5 GUI Element Specifications

This section contains the detailed specifications of the standard GUI elements that are defined by the panel subunit model. This detailed information is embedded in the GUI element, i.e. *element_data* field of the element data block, as described in section7.3.

The GUI element type value is specified in section 8.1.1 and 8.1.2

In each GUI element structure, the optional link list and optional attributes list, specified in section 10.3.3.1 and 10.4.3.1, are optional, the other attributes and links is mandatory except element link [x] fields in PANEL or GROUP. These mandatory links and attributes are described in section 10.3.2 and 10.4.2.

10.5.1 PANEL

The element_data field for a PANEL is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (09_{16})
00 03_{16}	panel_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	panel_aspect_ratio
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	number_of_mandatory_links (n)
00 $0D_{16}$	panel_label_link
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	element_link[1]
:	
:	
:	
:	:
:	element_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.53 – PANEL structure

A PANEL can contain GROUPs and non-organizational GUI elements but it cannot contain the other PANEL. A HELP PANEL and ALERT PANEL has the same structure as the PANEL, but the panel link to and from these panels might not exist. In this case, the controller that request the HELP PANEL or ALERT

PANEL to the target should display back to the previous PANEL after it erases the HELP PANEL or ALERT PANEL.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 09_{16}.

The *panel_width_height* field specifies the height and width, in pixels, of the GUI element using a width height attribute structure specified in section 10.4.2.1. The size of each PANEL shall be smaller than or equal to the safety display area size, which is *GUI resolution* field value in GUI UPDATE command.

The *panel_aspect_ratio* field specifies the aspect ratio of the images in this PANEL and use aspect ratio attribute structure specified in section 10.4.2.3.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this PANEL. So this number is equal to the total number of element_link[x] fields plus one.

The *panel_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *element_link[x]* field is a link to the element using a *element link* structure specified in section 10.3.2.3.

The *size_of_optional_attributes* field specifies the size of optional attributes. A PANEL can have zero or more optional attribute. In case of no optional attribute in the PANEL, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A PANEL can have zero or more optional links. In case of no optional link in the PANEL, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

Note that a PANEL is not interactive GUI element, and the controller will pull the panel data by using PUSH GUI DATA control command.

10.5.2 GROUP

The element_data field for a GROUP is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (06_{16})
00 03_{16}	group_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	number_of_mandatory_links (n)
00 $0A_{16}$	group_label_link
00 $0B_{16}$	
00 $0C_{16}$	
00 $0D_{16}$	
00 $0E_{16}$	element_link[1]
:	
:	
:	
:	:
:	element_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.54 – GROUP structure

A GROUP is used to suggest to the controller that the elements linked from the GROUP should be displayed together on the screen. This GROUP is useful when the controller has to scale down the panel. The GROUP cannot contain panel and the other GROUP.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 06_{16}.

The *group_width_height* field specifies the height and width, in pixels, of the GUI element using a width height attribute structure specified in section 10.4.2.1. The size of each GROUP shall be smaller than or equal to the organizational panel size.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this GROUP. So this number is equal to the total number of element_link[x] fields plus one.

The *group_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *element_link[x]* field is a link to the element using a *element link* structure specified in section 10.3.2.3.

The *size_of_optional_attributes* field specifies the size of optional attributes. A GROUP can have zero or more optional attribute. In case of no optional attribute in the GROUP, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A GROUP can have zero or more optional links. In case of no optional link in the GROUP, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

Note that a GROUP is not interactive GUI element, and the controller will pull the group data by using PUSH GUI DATA control command.

10.5.3 PANEL LINK

The element_data field for a PANEL LINK GUI element is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes ($0F_{16}$)
00 03_{16}	link_ width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	link_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	linked_panel
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	
00 12_{16}	number_of_mandatory_links (02_{16})
00 13_{16}	link_label_link
00 14_{16}	
00 15_{16}	
00 16_{16}	
00 17_{16}	link_image_link
00 18_{16}	
00 19_{16}	
00 $1A_{16}$	
00 $1B_{16}$	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.55 – PANEL LINK structure

A PANEL LINK is used to indicate to the controller which panel should be pulled when the user selects this PANEL LINK. Note that a PANEL LINK doesn't imply any transmission of a panel and any changes

of the current panel. The panel data transmission and current panel change are executed by using PUSH GUI DATA control command.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is $0F_{16}$.

The *link_width_height* field specifies the height and width, in pixels, of the GUI element specified in section 10.4.2.1.

The *link_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2. Note that the user can select the PANEL LINK even if interactive attribute is set to "display-only" and it is a trigger to get the linked panel by using PUSH GUI DATA control command.

The *linked_panel* field specifies the element_ID of the panel which should be shown to the user if this PANEL LINK element is selected. The *linked_panel* field uses a *linked panel ID* attribute structure defined in section 10.4.2.4.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this GUI element. In case of PANEL LINK, the value of this field is 02_{16}.

The *link_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *link_image_link* field is an *image link* structure specified in section 10.3.2.2.

The *size_of_optional_attributes* field specifies the size of optional attributes. A PANEL LINK can have zero or more optional attribute. In case of no optional attribute in the PANEL LINK, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A PANEL LINK can have zero or more optional links. In case of no optional link in the PANEL LINK, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case that the PANEL LINK is display-only indicated by link_interactive field, when the user can interact with a PANEL LINK element but the controller shall not send "selected" user action to the panel subunit. The controller manages the "navigation" from the current panel to a different panel, and it will fetch the next panel as indicated by the linked_panel attribute using PUSH GUI DATA command.

The PANEL LINK may be interactive indicated by link_interactive field. When the user interacts with a PANEL LINK element, the controller sends "selected" user action message to the panel subunit. But the controller manages the "navigation" from the current panel to a different panel, and it will fetch the next panel as indicated by the linked_panel attribute using PUSH GUI DATA command. The user selecting the PANEL LINK element does not result in request the linked panel data to the target other than the controller itself requesting the panel by using the PUSH GUI DATA command.

In case that the PANEL LINK is temporary non-interactive indicated by the link_interactive field, the user cannot interact with a PANEL LINK element.

10.5.4 BUTTON

The element_data field for a BUTTON is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (09_{16})
00 03_{16}	button_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	button_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	number_of_mandatory_links (04_{16})
00 $0D_{16}$	released_label_link
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	released_bitmap_link
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	pressed_label_link
00 16_{16}	
00 17_{16}	
00 18_{16}	
00 19_{16}	pressed_bitmap_link
00 $1A_{16}$	
00 $1B_{16}$	
00 $1C_{16}$	
00 $1D_{16}$	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.56 – BUTTON structure

A BUTTON is used to display two-state button, i.e. released and pressed state.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 09_{16}.

The *button_width_height* field specifies the height and width, in pixels, of the GUI element specified in section 10.4.2.1.

The *button_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this BUTTON.

The *released_label_link* and *pressed_label_link* fields use a *label link* structure specified in section 10.3.2.1.

The *released_bitmap_link* and *pressed_bitmap_link* fields use an *image link* structure specified in section 10.3.2.2.

The *size_of_optional_attributes* field specifies the size of optional attributes. A BUTTON can have zero or more optional attribute. In case of no optional attribute in the BUTTON, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A BUTTON can have zero or more optional links. In case of no optional link in the BUTTON, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

The *optional_link_list* field is a list of optional links to the element specified in section 10.3.3.1. When the *optional_link_list* doesn't exist, there is no *optional_link_list* field in this GUI element.

In case BUTTON is interactive, when the BUTTON has been pressed, then the controller plays pressed sound, changes the released bitmap to the pressed bitmap, and sends the USER ACTION command with press action specified in section 9.3.

When the BUTTON has been released, then the controller plays the release sound, changes the pressed bitmap to the released bitmap, and sends the USER ACTION command with release action specified in section 9.3.

In case that the BUTTON may not be interactive indicated by button_interactive field for a while, the user cannot interact with this BUTTON, e.g. the controller doesn't move the cursor to the BUTTON, and shall not send any user action of this GUI element to the target.

10.5.5 ANIMATION

The element_data field for an ANIMATION is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes ($0D_{16}$)
00 03_{16}	animation_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	animation_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	animation_type
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	number_of_mandatory_links (n)
00 11_{16}	animation_label_link
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	animation_element_text_link[1]
:	
:	
:	
:	animation_element_bitmap_link[2]
:	
:	
:	
:	:
:	animation_element_bitmap_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.57 – ANIMATION structure

An ANIMATION element has a set of multiple images, which is used for presenting an animation element, e.g. bitmap and text, of the ANIMATION.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is $0D_{16}$.

The *animation_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *animation_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *animation_type* field specifies the type of this GUI element, using an *animation type* attribute structure specified in section 10.4.2.5.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this ANIMATION. So this number is equal to the total number of animation_element_text_link[x] and animation_element_bitmap_link[x] fields plus one.

The *animation_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *animation_element_text_link[x]* field is an *animation element text link* defined in section 10.3.2.8.

The *animation_element__bitmap_link[x]* field is an *animation element bitmap link* defined in section 10.3.2.8.

The text and the bitmap specified in the next link field are displayed together at the same time.

One animation element shall have one text and one bitmap link for each animation element, but the link might be empty using $FFFF_{16}$. If either of them is empty, the controller doesn't display the empty link.

For the element number for USER ACTION command, one value is assigned to the set of animation element text and animation element bitmap. The value starts from one end to (n-1) / 2.

The *size_of_optional_attributes* field specifies the size of optional attributes. ANIMATION can have zero or more optional attribute. In case of no optional attribute in the ANIMATION, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. An ANIMATION can have zero or more optional links. In case of no optional link in the ANIMATION, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case that the ANIMATION is not interactive indicated by animation_interactive field, the user cannot interact with this ANIMATION, e.g. the controller doesn't move the cursor to the ANIMATION, and shall not send any user action of this GUI element to the target.

In case ANIMATION is interactive, when an animation element has been selected, then the controller plays select sound and sends the USER ACTION command with select_item action specified in section 9.3. The select_item action contains index number of the selected element, which assigned to animation elements,

i.e. the first animation element in ANIMATION structure is index number one, and the second one is index number two.

In case ANIMATION is interactive, repetition type is once playback and an animation element has not been selected, then the controller plays select sound and sends the USER ACTION command with the special action _data field value (FF_{16}) specified in section 9.3.

10.5.6 SHOW RANGE

The element_data field for a SHOW RANGE is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (14_{16})
00 03_{16}	range_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	range_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	range_type
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	
00 12_{16}	
00 13_{16}	value_set
00 14_{16}	
00 15_{16}	
00 16_{16}	
00 17_{16}	number_of_mandatory_links (01_{16})
00 18_{16}	range_label_link
00 19_{16}	
00 $1A_{16}$	
00 $1B_{16}$	
00 $1C_{16}$	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.58 – SHOW RANGE structure

A SHOW RANGE is used for displaying the range, e.g. audio level meter. The value of the SHOW RANGE is provided by the panel subunit, and the user cannot change the value of this element.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 14_{16}.

The *range_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *range_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *range_type* field specifies the type of this GUI element, using a *range type* attribute structure specified in section 10.4.2.6.

The *value_set* field specifies the current value of this GUI element, using a *value range* attribute structure specified in section 10.4.2.7.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this SHOW RANGE. So this number is one in a SHOW RANGE.

The *range_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *size_of_optional_attributes* field specifies the size of optional attributes. A SHOW RANGE can have zero or more optional attribute. In case of no optional attribute in the SHOW RANGE, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A SHOW RANGE can have zero or more optional links. In case of no optional link in the SHOW RANGE, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

The SHOW RANGE is the range initiated by the target, e.g. audio level meter. The controller cannot change the value of SHOW RANGE, specified in value_set field.

The target can send the current value, i.e. value_set field value, for the SHOW RANGE by using status report specified in section 7.9, whenever the current value has been changed by status changes in the target.

In case that the SHOW RANGE is not interactive indicated by range_interactive field, the user cannot interact with this SHOW RANGE, e.g. the controller doesn't move the cursor to the SHOW RANGE, and shall not send any user action of this GUI element to the target.

In case SHOW RANGE is interactive, when the SHOW RANGE has been selected, then the controller plays select sound and sends the USER ACTION command with select action specified in section 9.3.

Note that: the look and feel of this GUI element is controller dependent, so the controller can decide the detail shape of this GUI element.

10.5.7 SET RANGE

The element_data field for a SET RANGE is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (14_{16})
00 03_{16}	range_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	range_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	range_type
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	
00 12_{16}	
00 13_{16}	value_set
00 14_{16}	
00 15_{16}	
00 16_{16}	
00 17_{16}	number_of_mandatory_links (01_{16})
00 18_{16}	range_label_link
00 19_{16}	
00 $1A_{16}$	
00 $1B_{16}$	
00 $1C_{16}$	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.59 – SET RANGE structure

A SET RANGE is used to request the user to set the new value of the range, e.g. slider or dial.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 14_{16}.

The *range_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *range_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *range_type* field specifies the type of this GUI element, using a *range type* attribute structure specified in section 10.4.2.6.

The *value_set* field specifies the current value of this GUI element, using a *value range* attribute structure specified in section 10.4.2.7.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this SET RANGE. So this number is one in a SET RANGE.

The *range_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *size_of_optional_attributes* field specifies the size of optional attributes. A SET RANGE can have zero or more optional attribute. In case of no optional attribute in the SET RANGE, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A SET RANGE can have zero or more optional links. In case of no optional link in the SET RANGE, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

The SET RANGE is the range initiated by the controller, e.g. audio volume. The controller can request to change the current value of SET RANGE, specified in value_set field, using USER ACTION command with set_value action specified in section 9.3. The set_value action contains new value that the user want to change to, and the new value is sixteen bits unsigned numeric number.

The target can send the current value, i.e. value_set field value, for the SET RANGE by using status report specified in section 7.9, whenever the current value has been changed by status changes in the target, e.g. the target disagree on the user specified value, or someone operates the physical front panel of the device.

However, the SET RANGE may not be interactive for a while, indicated by range_interactive field, then the user cannot interact with this SET RANGE at this time, e.g. the controller doesn't move the cursor to the SET RANGE, and shall not send any user action of this GUI element to the target.

Note that: the look and feel of this GUI element is vendor dependent, so the controller can decide how the user set the new range value.

10.5.8 ENTRY

The element_data field for an ENTRY is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (17_{16})
00 03_{16}	entry_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	entry_interractive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	entry_type
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	entry_type_depend
00 11_{16}	
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	
00 16_{16}	
00 17_{16}	
00 18_{16}	
00 19_{16}	
00 $1A_{16}$	number_of_mandatory_links (01_{16})
00 $1B_{16}$	entry_label_link
00 $1C_{16}$	
00 $1D_{16}$	
00 $1E_{16}$	
00 $1F_{16}$	size_of_optional_attributes
00 20_{16}	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.60 – ENTRY structure

An ENTRY is used to request the user to input the value, text or date.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 17_{16}.

The *entry_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *entry_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *entry_type* field specifies the type of this GUI element, using an *entry type* attribute structure specified in section 10.4.2.8.

The *entry_type_depend* field specifies the type dependent information of this GUI element, using an *entry type depend* attribute structure specified in section 10.4.2.9.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this ENTRY. So this number is 01_{16} in an ENTRY.

The *entry_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *size_of_optional_attributes* field specifies the size of optional attributes. An ENTRY can have zero or more optional attribute. In case of no optional attribute in the ENTRY, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. An ENTRY can have zero or more optional links. In case of no optional link in the ENTRY, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

The target can send the new value as default value in entry_type_depend field or default text data entity by using GUI update mechanism, when the specified value has been changed by status changes in the target, e.g. someone operates the physical front panel of the device.

However, the ENTRY may not be interactive for a while, indicated by entry_interactive field, then the user cannot interact with this ENTRY at this time, e.g. the controller doesn't move the cursor to the ENTRY, and shall not send any user action of this GUI element to the target.

Note that: the look and feel of this GUI element is vendor dependent, so the controller can decide how the user inputs the data and how the user finishes the data input. For instance, the controller can display the "January" instead of "01" in month field of default date data, and the user can types "January" instead of "01" for the date entry. Furthermore the controller can display candidates for months in order for user to choose one of them.

The controller shall send the input data to the target by using USER ACTION control command, after the user finishes inputting the all data for the ENTRY.

10.5.8.1 A user action data format and a default data format for ENTRY

ENTRY has the data format used for both transmitting data in USER ACTION command and default data.

(1) Text entry

In case of text entry specified in the *entry_type* field, the intended data is text string and it is put in the action_data field in the USER ACTION command without any additional information. The data entity format is used for default strings if ENTRY has an optional *default text link* specified in section 10.3.3.2.4. If there is no default strings, the ENTRY shall not have optional *default text link*. The entry_type_depend field, defined in section 10.4.2.9, contains the maximum string size in bytes using a text entry string size data structure specified below.

byte count	msb							lsb
00_{16}	reserved							
01_{16}	reserved							
02_{16}	reserved							
03_{16}	reserved							
04_{16}	reserved							
05_{16}	reserved							
06_{16}	reserved							
07_{16}	entry_text_data_size							

Figure 10.61 – Text entry string size data structure

The *entry_text_data_size* field specifies the maximum size of the intended input data in bytes in case of text entry. The action_data field of the USER ACTION command for this ENTRY element is just string whose size is smaller than or equal to the size of the *entry_text_data_size* field.

(2) Numeric entry

In case of numeric entry specified in the *entry_type* field, the intended data is thirty-two bits unsigned numeric number and it is put in the action_data field of the USER ACTION command by using number value data structure specified below. This structure is also used in entry_type_depend field using *entry type depend* attribute defined in section 10.4.2.9

In case of default value, if the value of *entry type depend* attribute may be filled up with one then the controller should not display the default value. The ENTRY shall not have optional *default text link*.

byte count	msb							lsb
00_{16}	reserved							
01_{16}	reserved							
02_{16}	reserved							
03_{16}	reserved							
04_{16}	value							
05_{16}								
06_{16}								
07_{16}								

Figure 10.62 – Number value data structure

The *value* field specifies the value of the input data or default data for numeric or float entry.

(3) Float entry

In case of float entry specified in the *entry_type* field, the intended data is thirty-two bits float number and it is put in the action_data field of the USER ACTION command by using number value data structure. This structure is also used in entry_type_depend field using a *entry type depend* attribute specified in section 10.4.2.9 The ENTRY shall not have optional *default text link*.

(4) Date time entry

In case of date time entry specified in the *entry_type* field, the input date time value is in the action_data field of the USER ACTION command by using date time data structure below. The data_length field in the USER ACTION command shall be set to eight. This date time data structure is also used in entry_type_depend field using a *entry type depend* attribute specified in section 10.4.2.9 The ENTRY shall not have optional *default text link*.

The date time data is encoded as follows:

byte count	contents
00_{16}	year
01_{16}	
02_{16}	month
03_{16}	date
04_{16}	hour
05_{16}	minute
06_{16}	second
07_{16}	reserved

Figure 10.63 – Date time data structure

The *year*, *month*, *date*, *hour*, *minuets*, *second* fields are encoded as BCD format such as XXXX year, XX month, XX date, XX hour, XX minutes, XX seconds here each X shows the four bits BCD value.

When some of these fields in date time data structure may be $FFFF_{16}$ or FF_{16} then it means no default value for the fields.

When some of these fields in date time data structure may be $EEEE_{16}$ or EE_{16} then it is recommended that the controller doesn't display the default value of these fields and doesn't allow the user to input on these fields. The controller shall not change the value of these fields, i.e. the default values, and send all fields of date time structure to the target by user action.

The possible value of each field follows:

Table 10.8 – Possible value for date time

field	possible value (BCD code)
year	any four digits
month	01 to 12
date	01 to 31
hour	00 to 23
minute	00 to 59
second	00 to 59

10.5.9 CHOICE

The element_data field for a CHOICE is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes ($0D_{16}$)
00 03_{16}	choice_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	choice_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	choice_type
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	number_of_mandatory_links (n)
00 11_{16}	choice_label_link
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	choice_element_text_link[1]
:	
:	
:	
:	choice_element_bitmap_link[2]
:	
:	
:	
:	:
:	choice_element_bitmap_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.64 – CHOICE structure

A CHOICE is used to request the user to choose one or more candidates. These candidates should be displayed together if possible. The controller can use scroll capability for these candidates.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is $0D_{16}$.

The *choice_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *choice_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *choice_type* field specifies the type of this GUI element, using a *choice type* attribute structure specified in section 10.4.2.10.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this CHOICE. So this number is equal to the total number of choice_element_text_link[x] and choice_element_bitmap_link[x] fields plus one.

The *choice_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *choice_element_text_link[x]* field is a *choice element text link* defined in section 10.3.2.9.

The *choice_element__bitmap_link[x]* field is a *choice element bitmap link* defined in section 10.3.2.9.

The text and the bitmap specified in the next link field are displayed together.

One choice element shall have one text and one bitmap link for each choice element, but the link might be empty using $FFFF_{16}$. If either of them is empty, the controller doesn't display the empty link.

For the index number for USER ACTION command, one value is assigned to the set of choice element text and choice element bitmap. The value starts from one end to (n-1) / 2.

The *size_of_optional_attributes* field specifies the size of optional attributes. A CHOICE can have zero or more optional attribute. In case of no optional attribute in the CHOICE, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A CHOICE can have zero or more optional links. In case of no optional link in the CHOICE, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case CHOICE is interactive, when a choice elements have been selected, then the controller sends the USER ACTION command with choose_list action specified in section 9.3. The choose_list action contains index numbers of the selected elements, which assigned to choice elements, i.e. the first choice element in CHOICE structure is index number one, and the second one is index number two.

When the CHOICE is focused but no element is chosen, then the controller shall not send any user action.

In case that the CHOICE may not be interactive indicated by choice_interactive field for a while, the user cannot interact with this CHOICE, e.g. the controller doesn't move the cursor to the CHOICE, and shall not send any user action of this GUI element to the target.

Note that: the look and feel of this GUI element is vendor dependent, so the controller can decide how the user selects the element(s) and how the user finishes selecting the element(s).

10.5.10 TEXT

The element_data field for a TEXT is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (09_{16})
00 03_{16}	text_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	text_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	number_of_mandatory_links (01_{16})
00 $0D_{16}$	text_link
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.65 – TEXT structure

The TEXT is used to display the text strings and.it might be selectable.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 09_{16}.

The *text_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *text_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this TEXT. So this number is 01_{16} in a TEXT.

The *text_link* field is a *text link* structure specified in section 10.3.2.10.

The *size_of_optional_attributes* field specifies the size of optional attributes. A TEXT can have zero or more optional attribute. In case of no optional attribute in the TEXT, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A TEXT can have zero or more optional links. In case of no optional link in the TEXT, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case TEXT is interactive, when the TEXT has been selected, then the controller plays selected sound and sends the USER ACTION command with select action specified in section 9.3.

In case that the TEXT is not interactive indicated by text_interactive field, the user cannot interact with this TEXT, e.g. the controller doesn't move the cursor to the TEXT, and shall not send any user action of this GUI element to the target, because the TEXT is used for just information to the user.

10.5.11 STATUS

The element_data field for a STATUS is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes ($0C_{16}$)
00 03_{16}	status_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	status_type
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	status_working
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	number_of_mandatory_links (01_{16})
00 10_{16}	status_label_link
00 11_{16}	
00 12_{16}	
00 13_{16}	
00 14_{16}	size_of_optional_attributes
00 15_{16}	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.66 – STATUS structure

A STATUS is used to inform the user that the target is busy.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is $0C_{16}$.

The *status_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *status_type* field specifies the type of this GUI element, using a *status type* attribute structure specified in section 10.4.2.11.

The *status_working* field specifies the type of this GUI element, using a *status working* attribute structure specified in section 10.4.2.12.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this STATUS. So this number is 01_{16} in a STATUS.

The *status_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *size_of_optional_attributes* field specifies the size of optional attributes. A STATUS can have zero or more optional attribute. In case of no optional attribute in the STATUS, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A STATUS can have zero or more optional links. In case of no optional link in the STATUS, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

Note that: the look and feel of this GUI element is vendor dependent, so the controller can decide the detail shape of this GUI element.

10.5.12 ICON

The element_data field for an ICON is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (09_{16})
00 03_{16}	icon_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	icon_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	number_of_mandatory_links (02_{16})
00 $0D_{16}$	icon_label_link
00 $0E_{16}$	
00 $0F_{16}$	
00 10_{16}	
00 11_{16}	icon_bitmap_link
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	size_of_optional_attributes
00 16_{16}	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.67 – ICON structure

An ICON is used to display an image for either selection or display only. If the controller displays the image then it is suggested that it doesn't display the label.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 09_{16}.

The *icon_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *icon_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this ICON. So this number is 02_{16} in an ICON.

The *icon_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *icon_bitmap_link* field is an *image link* structure specified in section 10.3.2.2.

The ICON shall have one text and one bitmap link, but these links might be empty, i.e. the element_ID field in the link is set to $FFFF_{16}$, if the target doesn't want to display either of them.

The *size_of_optional_attributes* field specifies the size of optional attributes. An ICON can have zero or more optional attribute. In case of no optional attribute in the ICON, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. An ICON can have zero or more optional links. In case of no optional link in the ICON, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case ICON is interactive, when the ICON has been selected, then the controller plays selected sound and sends the USER ACTION command with select action specified in section 9.3.

In case that the ICON is not interactive indicated by icon_interactive field, the user cannot interact with this ICON, e.g. the controller doesn't move the cursor to the ICON, and shall not send any user action of this GUI element to the target, because the ICON is used for just information to the user.

10.5.13 Toggle

The element_data field for a TOGGLE is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes ($0C_{16}$)
00 03_{16}	toggle_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	toggle_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	toggle_status
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	number_of_mandatory_links (n)
00 10_{16}	toggle_label_link
00 11_{16}	
00 12_{16}	
00 13_{16}	
00 14_{16}	toggle_element_text_link[1]
:	
:	
:	
:	toggle_element_bitmap_link[2]
:	
:	
:	
:	:
:	toggle_element_bitmap_link[n-1]
:	
:	
:	
:	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.68 – TOGGLE structure

A TOGGLE is used for a choice from several toggle states and the controller should display show the only current state of the TOGGLE. The toggle state is changed to next state whenever the user selects this TOGGLE. When the user selects the TOGGLE in case of the last toggle state, the current toggle state is changed to the first state.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value is $0C_{16}$.

The *toggle_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *toggle_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *toggle_status* field specifies the current status of this GUI element, using a *toggle status* attribute structure specified in section 10.4.2.13.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this TOGGLE. So this number is equal to the total number of toggle_element_text_link[x] and toggle_element_bitmap_link[x] fields plus one.

The *toggle_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *toggle_element_text_link[x]* field is a *toggle element text link* defined in section 10.3.2.11.

The *toggle_element__bitmap_link[x]* field is a *toggle element bitmap link* defined in section 10.3.2.11.

The text and the bitmap specified in the next link field are displayed together.

One toggle element shall have one text and one bitmap link for each toggle element, but the link might be empty using $FFFF_{16}$.

For the index number for USER ACTION command, one value is assigned to the set of toggle element text and toggle element bitmap. The value starts from one end to (n-1) / 2. Refer to section 9.3

The *size_of_optional_attributes* field specifies the size of optional attributes. A TOGGLE can have zero or more optional attribute. In case of no optional attribute in the TOGGLE, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A TOGGLE can have zero or more optional links. In case of no optional link in the TOGGLE, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case the TOGGLE is interactive, when the TOGGLE has been selected, then the controller plays select sound, changes the surface of the TOGGLE to the bitmap of the next toggle element, and sends the USER ACTION command with select action whenever it is selected. The controller will change the toggle element on the screen to next element in the listed order in TOGGLE structure. Further the controller displays the first one when the user select the TOGGLE after it reaches the end of toggle element.

Regardless whether the panel subunit permit changing the state or not, the panel subunit should send the new state value to the controller via asynchronous connection using status report mechanism. For detail, refer to section 7.9. The controller notices the exact current state by the status report even if the panel subunit doesn't allow the state change.

In case that the TOGGLE may not be interactive indicated by toggle_interactive field for a while, the user cannot interact with this TOGGLE at this time, e.g. the controller doesn't move the cursor to the TOGGLE, and shall not send any user action of this GUI element to the target.

10.5.14 CONTENT ICON

The element_data field for a CONTENT ICON is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (16_{16})
00 03_{16}	content_width_height
00 04_{16}	
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	
00 09_{16}	content_interactive
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	content_availability
00 $0D_{16}$	
00 $0E_{16}$	
00 $0F_{16}$	content_plug
00 10_{16}	
00 11_{16}	
00 12_{16}	
00 13_{16}	
00 14_{16}	
00 15_{16}	
00 16_{16}	
00 17_{16}	
00 18_{16}	
00 19_{16}	number_of_mandatory_links (02_{16})
00 $1A_{16}$	content_label_link
00 $1B_{16}$	
00 $1C_{16}$	
00 $1D_{16}$	
00 $1E_{16}$	content_bitmap_link
00 $1F_{16}$	
00 20_{16}	
00 21_{16}	
00 22_{16}	size_of_optional_attributes
:	list_of_optional_attribute
:	
:	
:	number_of_optional_links (m)
:	optional_link[0]
:	
:	
:	
:	:
:	optional_link[m-1]
:	
:	
:	

Figure 10.69 – CONTENT ICON structure

A CONTENT ICON is used for selection and display of the contents that the target has.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 16_{16}.

The *content_width_height* field specifies the height and width, in pixels, of the GUI element, using a width height attribute structure specified in section 10.4.2.1.

The *content_interactive* field specifies the current interactivity of this GUI element, using an *interactive* attribute structure specified in section 10.4.2.2.

The *content_availability* field specifies the current status of this GUI element, using a *content availability* attribute structure specified in section 10.4.2.14.

The *content_plug* field specifies the input or output plug of this GUI element, using a *AV plug* attribute structure specified in section 10.4.2.15.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this CONTENT ICON. So this number is 02_{16} in a CONTENT ICON.

The *content_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *content_bitmap_link* field is an *image link* structure specified in section 10.3.2.2.

One CONTENT ICON shall have one text and one bitmap link, but the link might be empty using $FFFF_{16}$.

The *size_of_optional_attributes* field specifies the size of optional attributes. A CONTENT ICON can have zero or more optional attribute. In case of no optional attribute in the CONTENT ICON, this field value is zero and no *list_of_optional_attribute* field exists.

The *list_of_optional_attribute* field contains zero or more optional attributes, as specified in section 10.4.3.1, and the possible optional attributes are specified in Table 10.2 – Optional info_type field value.

The *number_of_optional_links* field contains the total number of optional links. A CONTENT ICON can have zero or more optional links. In case of no optional link in the CONTENT ICON, this field value is zero and no *optional_link* field exists.

The *optional_link[x]* field is a optional links specified in section 10.3.3.1, and the possible optional links are specified in Table 10.2 – Optional info_type field value.

In case CONTENT ICON is interactive, when the CONTENT ICON has been selected, then the controller plays selected sound and sends the USER ACTION command with select action specified in section 9.3.

Whenever the target receives the select user action, the target will try to invert the *working* bit in the *content availability* field specified in section 10.4.2.14. After that if the target cannot invert the *working* bit, then the target will try to invert *schedule* bit in the *content_availability* field.

For instance, in case the indicated content is available and not outputted, i.e. the *working* bit is zero, if the user select this GUI element, then target will change the *working* bit to one and start outputting the indicated program on the specified plug by *AV plug* attribute.

In case the indicated content is not available, not outputted but CONTENT ICON has the *broadcast date* attribute, if the user select this GUI element, then target will try to change the *schedule* bit to one and start outputting the indicated content on the specified plug by *AV plug* attribute on the specified time by *broadcast date* attribute.

Both *working* bit and *schedule* bit shall not be set to one at the same time.

In case either *working* bit or *schedule* bit is set to one, if the user select this GUI element, then target will try to change the bit, which is set to one, to zero.

In case that the CONTENT ICON may not be not interactive indicated by *content_interactive* field, the user cannot interact with this CONTENT ICON, e.g. the controller doesn't move the cursor to the CONTENT ICON, and shall not send any user action of this GUI element to the target.

10.5.15 DEVICE ICON

The element_data field for a DEVICE ICON is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (00_{16})
00 03_{16}	number_of_mandatory_links (02_{16})
00 04_{16}	device_label_link
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	device_bitmap_link
00 09_{16}	
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	size_of_optional_attributes (00_{16})
00 $0D_{16}$	number_of_optional_links (00_{16})

Figure 10.70 – DEVICE ICON structure

A DEVICE ICON is used for selection and display of the target device itself, and it might be module icon or modifiable icon. The target should use the best icon, e.g. device icon, module icon or modifiable icon, to represent itself as DEVICE ICON.

The *size_of_mandatory_atributes* field specifies the size of mandatory attributes, and the value of this field is 00_{16} because there is no mandatory attribute for a DEVICE ICON.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this DEVICE ICON. So this number is 02_{16} in a DEVICE ICON.

The *device_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *device_bitmap_link* field is an *image link* structure specified in section 10.3.2.2.

The *size_of_optional_attributes* field specifies the size of optional attributes. A DEVICE ICON has zero optional attributes, so this field value is zero.

The *number_of_optional_links* field contains the total number of optional links. A DEVICE ICON has zero optional links, so this field value is zero.

The size of bitmap should be 48 x 48 pixels for consistency.

10.5.16 VENDOR ICON

The element_data field for a VENDOR ICON is as follows:

byte count	contents
00 00_{16}	element_length
00 01_{16}	
00 02_{16}	size_of_mandatory_attributes (00_{16})
00 03_{16}	number_of_mandatory_links (02_{16})
00 04_{16}	vendor_label_link
00 05_{16}	
00 06_{16}	
00 07_{16}	
00 08_{16}	vendor_bitmap_link
00 09_{16}	
00 $0A_{16}$	
00 $0B_{16}$	
00 $0C_{16}$	size_of_optional_attributes (00_{16})
00 $0D_{16}$	number_of_optional_links (00_{16})

Figure 10.71 – VENDOR ICON structure

A VENDOR ICON is used for display of the target vendor, e.g. logo of the vendor.

The *size_of_mandatory_attributes* field specifies the size of mandatory attributes, and the value of this field is 00_{16} because there is no mandatory attribute for a VENDOR ICON.

The *number_of_mandatory_links* field contains the total number of mandatory links that linked from this VENDOR ICON. So this number is 02_{16} in a VENDOR ICON.

The *vendor_label_link* field is a *label link* structure specified in section 10.3.2.1.

The *vendor_bitmap_link* field is an *image link* structure specified in section 10.3.2.2.

The *size_of_optional_attributes* field specifies the size of optional attributes. A VENDOR ICON has zero optional attributes, so this field value is zero.

The *number_of_optional_links* field contains the total number of optional links. A VENDOR ICON has zero optional links, so this field value is zero.

The size of bitmap should be 48 x 48 pixels for consistency.

This page is left intentionally blank

Annexes

Annex A. Panel subunit controller presentation guidelines (informative)

A panel subunit controller has full ownership for presentation of panel data from a panel subunit. However, the guidelines mentioned below are recommended in order to keep some presentation consistency among panel subunit controllers made by different companies. These guidelines are, however, only recommendations and application software can create different presentations, using the panel data elements as hints.

Presentation Guideline

The general presentation guideline placed on a panel subunit controller is to:

- Display (using a controller-specific rendering, given its user input/output capabilities) the information contained in the attributes of the GUI elements that the controller has pulled from the target. Note, this requirement is only specified abstractly: the controller should give a "best-effort" (relative to its actual capabilities) presentation of the information embodied in the elements; however, no particular form or level of rendering is required.
- It is however recommended that the controller makes it possible for a user to see all GUI elements of a panel, e.g. by using scrolling facilities. It is also recommended that a controller makes it possible for a user to access all reachable panels (those for which the target supplied panel links).
- Allow the user to manipulate in a controller-specific, best-effort manner any interactive GUI element that the controller has pulled from the target so as to result in the generation of any action (with any attribute values that the action is capable of having) associated with that interactive element. Note, again this requirement is expressed abstractly with respect to the particular user-controller interaction that the controller uses to allow the user to: choose an interactive element, specify the attributes of an action, and indicate that the action should be sent by the controller to the target.
- If a controller shows a GUI element to the user, it should be the most recent, correct version (not an old version). Therefore the controller has to check GUI updates from the target, and get new GUI element(s). If the controller receive a suggested new panel identifier, it should pull the new panel and display it if possible.
- All elements (except the TEXT element which already contains a text string) have a mandatory label attribute that is defined to be a text string of length between zero and 16 characters. These labels have to be shown by the controller if it is not empty.

General Presentation Recommendations

Presentation recommendations for panel subunit controllers allow for two types of scaling: panel scaling and element scaling. Both types of scaling are explained below.

Panel Scaling

A panel element is intended to represent an entire screen's worth of GUI elements. Within the panel can be zero or more groups, each of which may contain individual elements. In addition, a panel can have standalone elements that are not part of any group.
For controllers showing several panels on their screen, the navigation between these panels is proprietary (there are however recommendations with respect to usage of panel links).
A controller can present a given panel in one of three styles: full-capability, intermediate, and basic.

- If the controller's panel is displayed on the screen as required by all elements supplied by the target then the controller is said to be operating in "full-capability".
- If the controller cannot display the entire panel as given it can then drop down to the next style of "intermediate". Suggestions on how to scale down would be represented by the target in its placement of elements within groups and groups within a panel. The intermediate controller could then display one group at a time and allow the user to locally navigate between groups by displaying next and previous groups.
- The order in which the elements and groups of a panel should be shown is determined by their order inside the panel/group. It is recommended that controllers make sure that the user knows the hierarchical level of the shown group within the panel. E.g. by showing (as much as possible) the labels of the other elements or groups.
- The last style, "basic", would be used if the controller can only display a few or one GUI element at a time. The basic controller could then display a set of elements at a time and allow the user to locally navigate between elements.
- The order in which the elements should be shown is determined by their order inside the panel/group. It is recommended that controllers make sure that the user knows the hierarchical level of the shown elements within the panel. E.g. by showing (as much as possible) the labels of the other elements or groups.

Which of the above styles the controller chooses depends on its capability and also whether the screen is concurrently used for other purposes (e.g. video rendering).
Note that the capabilities that the target assumes for element sizes and position might clash with the controller's capabilities for these attributes. This can still make it impossible for a controller that is in principle "full-capability", to show all the elements of the panel together.

Element Scaling

As given in General Presentation guideline for panel subunit controllers, a controller should display information contained in GUI elements. However, the controller has the freedom to choose the particular rendering and user interaction operation. Especially for non-organizational elements, the controller can choose between different rendering possibilities.

- All GUI element attributes are intended to be suggestions to the controller on how to organize and present the target's panels, groups, and elements to the user. However a controller has to take into account the GUI element attributes with best effort. In case the controller does not care about it, the controller can neglect the attributes.
- Ideally GUI elements should be positioned on the panel or group according to the (optional) position attribute supplied by the target. If this is not possible (not supplied by target, basic panel style chosen, etc.), the controller is recommended to render the elements on the screen in the order supplied by the target in either horizontal or vertical direction.
- It is required that, if targets supply position attributes, they do this for all elements of the group/panel. Note that targets should not rely on the precise position of the elements that they specify. The target should use one icon (bitmap) that graphically contains some surfaces of GUI elements if it keeps the precise position.

The above guidelines allow controllers to use rather different styles of representing GUI elements by ignoring or adapting certain types of attributes. It is recommended that a controller use the same style for all the elements of the same panel or group.

Annex B. Resource limitation for GUI elements (informative)

Safety parameter size limitation

The following table shows the safety parameter size limitations for GUI elements and attributes.

Table B.1 – Safety parameter size limitation

item	safety parameter size limitation
element_links in panel and group	256
bitmap	no limitation*[1]
sound	no limitation*[1]
labels	16 characters
text_link	1024 characters
hotlink	256 characters
animation_element_links	32 sets of elements(text and bitmap)
choice_element_links	64 sets of elements(text and bitmap)
toggle_element_links	32 sets of elements(text and bitmap)
optional_links_list	16 optional links
list_of_optional_attribute	16 optional attributes

*[1]: If the controller doesn't have enough resource then it can discard the data and may use the text data for example.

Annex C. Application example of function keys (informative)

This annex is strictly for the purposes of showing assumed examples that may apply function keys of PASS THROUCH command. Note that any information described in this annex is not intended to apply for actual implementation.

Figure C.1 shows an example of bus configuration that application examples of the function key operation are assumed. A user operates on Monitor, a controller of panel subunit, by using a remote commander. Figure C.2 shows an example of a remote commander on which function keys are mounted physically. When Monitor is in controlling one of other devices that support panel subunit, it converts a user operation to PASS THROUGH command and transmits to the device. Function key operation is available to any of the devices. Actions assigned to the function keys may be different by each device and may be previously set by the manufacturer or customised by users. Those actions may be presented to the user through OSD or described in the users manual.

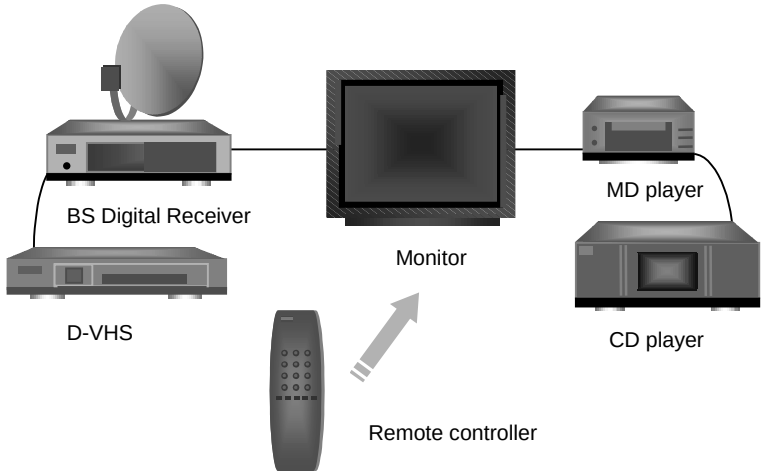

Figure C. 1 – Example bus configuration

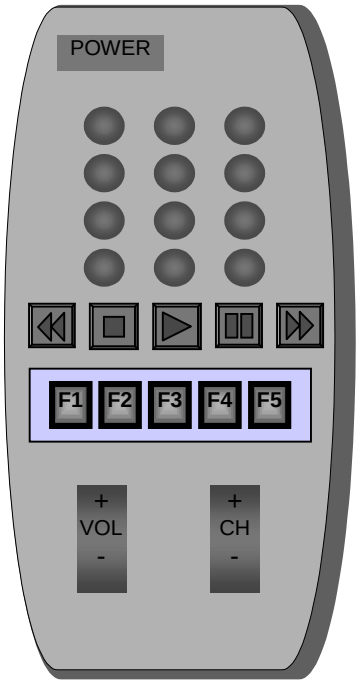

Figure C. 2 – Remote commander with function keys

Features of specific device type

– Color keys for Teletext

The application of Teletext System is defined in ETS 300 706 [R16], which is designed for broadcast media using cable, terrestrial and satellite transmission. A decoder provides four keys on its remote commander, associated with the first four linked page addresses, in transmission order. These keys shall be respectively coloured red, green, yellow and cyan and related to the associated displayable prompts.

The associated actions for the four keys above may be assigned to the function keys of PASS THROUGH command on the decorder when it supports panel subunit. Table below shows an example of function key assignment.

Table C.1 – Color button for Teletext decorder (example only)

Function key	Feature	Operation
F1	Red	Selects option that is shown related to red button
F2	Green	Selects option that is shown related to green button
F3	Yellow	Selects option that is shown related to yellow button
F4	Cyan	Selects option that is shown related to cyan button

These four keys are also used for EPG control, which may be applied to these function keys as well.

– Color keys for BS digital tuner

BS digital broadcast system in Japan provides the data broadcasting service, which presents the user interface that lets users select contents from the menu or messages displayed on the screen. In some applications, four color keys may appear on the screen.

The color keys of blue, red, green and yellow are defined in ARIB TR-B15 [R17]. The associated actions for the color keys may be assigned to the function keys of PASS THROUGH command on the BS digital tuner when it supports panel subunit. Table below shows an example of function key assignment.

Table C.2 – Color button for BS digital tuner (example only)

Function key	Feature	Operation
F1	Blue	Selects option that is shown related to blue button
F2	Red	Selects option that is shown related to red button
F3	Green	Selects option that is shown related to green button
F4	Yellow	Selects option that is shown related to yellow button

Device dependent actions

Function keys may be assigned to unique actions that is device dependent. Function assignment may be arranged at the manufacturer, or let users to customize according to their preferences.

– Short-cut operation

In order to operate some features, a user may have to open the menu and operate through the hierarchical command structure. For short-cut, the function keys may be assigned to such features. Table below shows the example of function key assignment for short-cut operation.

Table C.3 – VCR as a timer (example only)

Function key	Feature	Operation
F1	Timer on/off	Turn on or off the wakeup timer, which is set previously
F2	Sleep timer	Turn off the power automatically at specified time

– Device unique action

Device unique actions can be assigned with function keys. For example, when a user sets ‘repeat’ and ‘shuffle’ actions to the function keys on a CD player, function keys via PASS THROUGH commands work for repeat and shuffle actions.

Table C.4 – CD player (example only)

Function key	Feature	Operation
F1	Repeat	Repeat all songs of the inserted CD
F2	Shuffle	Shuffle the order of songs of the inserted CD to playback