COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

ARINC SPECIFICIATION 661-3

PUBLISHED: November 15, 2007

AN ARINC DOCUMENT

Prepared by
AIRLINES ELECTRONIC ENGINEERING COMMITTEE
Published by
AERONAUTICAL RADIO, INC.
2551 RIVA ROAD, ANNAPOLIS, MARYLAND 21401-7435

DISCLAIMER

THIS DOCUMENT IS BASED ON MATERIAL SUBMITTED BY VARIOUS PARTICIPANTS DURING THE DRAFTING PROCESS. NEITHER AEEC NOR ARINC HAS MADE ANY DETERMINATION WHETHER THESE MATERIALS COULD BE SUBJECT TO VALID CLAIMS OF PATENT, COPYRIGHT OR OTHER PROPRIETARY RIGHTS BY THIRD PARTIES, AND NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, IS MADE IN THIS REGARD.

AEEC USES REASONABLE EFFORTS TO DEVELOP AND MAINTAIN THESE DOCUMENTS. HOWEVER, NO CERTIFICATION OR WARRANTY IS MADE AS TO THE TECHNICAL ACCURACY OR SUFFICIENCY OF THE DOCUMENTS, THE ADEQUACY, MERCHANTABILITY, FITNESS FOR INTENDED PURPOSE OR SAFETY OF ANY PRODUCTS, COMPONENTS, OR SYSTEMS DESIGNED, TESTED, RATED, INSTALLED OR OPERATED IN ACCORDANCE WITH ANY ASPECT OF THIS DOCUMENT OR THE ABSENCE OF RISK OR HAZARD ASSOCIATED WITH SUCH PRODUCTS, COMPONENTS, OR SYSTEMS. THE USER OF THIS DOCUMENT ACKNOWLEDGES THAT IT SHALL BE SOLELY RESPONSIBLE FOR ANY LOSS, CLAIM OR DAMAGE THAT IT MAY INCUR IN CONNECTION WITH ITS USE OF OR RELIANCE ON THIS DOCUMENT, AND SHALL HOLD ARINC, AEEC, AND ANY PARTY THAT PARTICIPATED IN THE DRAFTING OF THE DOCUMENT HARMLESS AGAINST ANY CLAIM ARISING FROM ITS USE OF THE STANDARD.

THE USE IN THIS DOCUMENT OF ANY TERM, SUCH AS SHALL OR MUST, IS NOT INTENDED TO AFFECT THE STATUS OF THIS DOCUMENT AS A VOLUNTARY STANDARD OR IN ANY WAY TO MODIFY THE ABOVE DISCLAIMER. NOTHING HEREIN SHALL BE DEEMED TO REQUIRE ANY PROVIDER OF EQUIPMENT TO INCORPORATE ANY ELEMENT OF THIS STANDARD IN ITS PRODUCT. HOWEVER, VENDORS WHICH REPRESENT THAT THEIR PRODUCTS ARE COMPLIANT WITH THIS STANDARD SHALL BE DEEMED ALSO TO HAVE REPRESENTED THAT THEIR PRODUCTS CONTAIN OR CONFORM TO THE FEATURES THAT ARE DESCRIBED AS MUST OR SHALL IN THE STANDARD.

ANY USE OF OR RELIANCE ON THIS DOCUMENT SHALL CONSTITUTE AN ACCEPTANCE THEREOF “AS IS” AND BE SUBJECT TO THIS DISCLAIMER.

ARINC SPECIFICATION 661-3

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

Published: November 15, 2007

Prepared by the AEEC

Specification 661	Adopted by the Airlines Electronic Engineering Executive Committee	November 12, 2001

Summary of Document Supplements

Supplement	Adoption Date	Published
Specification 661-1	March 6, 2003	June 26, 2003
Specification 661-2	October 27, 2004	June 30, 2005
Specification 661-3	September 26, 2007	November 15, 2007

A description of the changes introduced by each supplement is included on Goldenrod paper at the end of this document.

FOREWORD

Aeronautical Radio, Inc., the AEEC and ARINC Standards

Aeronautical Radio, Inc. (ARINC) was incorporated in 1929 by four fledgling airlines in the United States as a privately-owned company dedicated to serving the communications needs of the air transport industry.

ARINC sponsors aviation industry committees and participates in related industry activities that benefit aviation at large by providing technical leadership and guidance and frequency management. These activities directly support airline goals: promote safety, efficiency, regularity, and cost-effectiveness in aircraft operations.

The AEEC is an international body of airline technical professionals that leads the development of technical standards for airborne electronic equipment-including avionics and in-flight entertainment equipment-used in commercial, military, and business aviation. The AEEC establishes consensus-based, voluntary form, fit, function, and interface standards that are published by ARINC and are known as ARINC Standards. The use of ARINC Standards results in substantial benefits to airlines by allowing avionics interchangeability and commonality and reducing avionics cost by promoting competition.

There are three classes of ARINC Standards:

a) ARINC Characteristics – Define the form, fit, function, and interfaces of avionics and other airline electronic equipment. ARINC Characteristics indicate to prospective manufacturers of airline electronic equipment the considered and coordinated opinion of the airline technical community concerning the requisites of new equipment including standardized physical and electrical characteristics to foster interchangeability and competition.

b) ARINC Specifications – Are principally used to define either the physical packaging or mounting of avionics equipment, data communication standards, or a high-level computer language.

c) ARINC Reports – Provide guidelines or general information found by the airlines to be good practices, often related to avionics maintenance and support.

The release of an ARINC Standard does not obligate any airline or ARINC to purchase equipment so described, nor does it establish or indicate recognition or the existence of an operational requirement for such equipment, nor does it constitute endorsement of any manufacturer’s product designed or built to meet the ARINC Standard.

In order to facilitate the continuous product improvement of this ARINC Standard, two items are included in the back of this volume:

An Errata Report solicits any corrections to the text or diagrams in this ARINC Standard.

An ARINC IA Project Initiation/Modification (APIM) form solicits any recommendations for addition of substantive material to this volume which would be the subject of a new supplement.

1.0 INTRODUCTION

1.1 Purpose and Scope

This document defines a standard Cockpit Display System (CDS) interface intended for all types of aircraft installations. The primary objective is to minimize the cost to the airlines, directly or indirectly by accomplishing the following:

- Minimize the cost of acquiring new avionic systems to the extent it is driven by the cost of CDS development
- Minimize the cost of adding new display function to the cockpit during the life of an aircraft
- Minimize the cost of managing hardware obsolescence in an area of rapidly evolving technology
- Introduce interactivity to the cockpit, thus providing a basis for airframe manufacturers to standardize the Human Machine Interface (HMI) in the cockpit

This document defines two external interfaces between the CDS and the aircraft systems. The first is the interface between the avionics equipment (user systems) and the display system graphics generators. The second is a means by which symbology and its related behavior is defined. A user application is defined as a system that transmits data to the CDS, which, in turn can be displayed as visual graphical information to the flight deck crew. A user application can also include software or hardware that receives input from interactive graphics managed by the CDS.

The CDS provides graphical and interactive services to user applications within the flight deck environment. When combined with data from user applications, it should display graphical images to the flight deck crew.

This document defines an interface between the CDS and user applications (UA). The application that controls the interface is defined to be within the CDS.

This document does not specify the “look and feel” of any graphical information.

1.2 Relationship to Other Documents

ARINC Specification 661 defines an interface protocol intended to facilitate communication between the cockpit display system and user equipment. This document does not specify electrical parameters.

This document refers to user application data formats that are specified in existing ARINC 700-series documents:

ARINC Characteristic 702A: *Advanced Flight Management Computer System*

ARINC Characteristic 708A: *Airborne Weather Radar with Forward Looking Windshear Detection Capability*

ARINC Characteristic 735A: *Traffic Alert and Collision Avoidance System (TCAS)*

ARINC Characteristic 762: *Terrain Awareness and Warning System (TAWS)*

Communication between user applications and the cockpit display system may be implemented over a physical data bus defined in system-level standards, such as:

ARINC Specification 429: *Mark 33 Digital Information Transfer System (DITS)*

AEEC Project Paper 453: *Very High Speed (VHS) Bus*

ARINC Specification 664: *Aircraft Data Network*

1.3 Interoperability

1.3.1 General

One of the primary objectives of this document is to define interface protocols that can be met by any equipment manufacturer. This level of interface standardization is different from a typical form, fit, and function standard.

This document emphasizes the need for standardized communication between the CDS and user applications. This approach is expected to facilitate the development of standardized subsystems that can easily interface with the CDS.

COMMENTARY

It is not the intention of this Specification to specify a bus structure, either physically or electrically. However, airlines encourage display system providers and aircraft system integrators to use industry standard buses. Manufacturers should recognize the practical advantages of developing equipment in accordance with the standards set forth in this document.

This document is not intended to define CDS packaging or physical configuration. It is noted, however, that some designs may be more suitable than others for use in a flight deck.

Avionics user systems should connect to the cockpit display system using interfaces based upon industry standards. This allows flexibility in the installation with the wide variety of display systems.

The desire for interoperability makes it necessary to standardize input and output interface parameters. The CDS interfaces should be capable of exchanging data in the form of input/output messages as defined in this Specification.

1.3.2 Interface Standards

In recognition of the widely varying cockpit layouts and configurations, standardization of equipment is not included within this standard.

This Specification defines a set of logical interfaces that support change containment and preserve investment across both aircraft types and hardware generations.

COMMENTARY

It is widely recognized that software qualification and system certification costs dwarf all other aspects of developing, installing, and updating a CDS.

1.3.3 Modularity

Architecturally, the CDS should be an integrated system comprised of modular hardware and software components. It should be possible to include optional features to individual units, as determined by airline user requirements, with minimum impact on the existing functions.

This Specification emphasizes that software necessary to add or change display functionality of the display system should be contained within the user application (e.g., FMS). Thus, during system upgrade or modification, only the user application software should need to change.

COMMENTARY

Airlines, airframe manufacturers, display system providers and user application developers contributed to the development of this Specification. The application developers advocated strict adherence to the preceding paragraph. However, others express caution that this proposal is not feasible from a certification standpoint. The writers of this document supported a compromise, which are detailed in later sections of this document.

User application developers should consider the role of a Cursor Control Device (CCD) in their equipment design. Application software should be structured in a manner that allows addition or modification of ARINC 661 input in general and cursor-based commands in particular. Software should be capable of adapting the HMI to fit different cockpit philosophies. This is expected to evolve as airline crews gain experience with existing and evolving levels of interactivity.

1.4 Integrity and Availability

The CDS is a significant portion of the flight deck crew interface. Therefore, the equipment design should contribute positively to overall aircraft system performance, operational integrity and availability goals for all types of aircraft operations.

1.5 Reliability

The airlines desire reliability in all phases in the design, production, installation and operation of a display system.

COMMENTARY

This document does not specify reliability goals. As a general rule, users want all they can get within the bounds of reasonable equipment complexity and cost.

1.6 Use of "Specification Language"

The vast majority of military and government standards are usually written in terms of "shall" and "shall not." However, it is often difficult to describe airline operator preferences that have grown out of experience over time. For this reason, this Specification is written to express airline desires in the form of guidance material. Designers should interpret this document in terms of the "need" for specific design practices rather than practices that "must" be met under all circumstances.

1.7 Regulatory Approval

ARINC 661 display equipment should meet all applicable regulatory requirements. This Specification should not and does not define the specific requirements that an equipment manufacturer must follow to be assured of approval. Such information should be obtained from the appropriate regulatory authority.

1.8 Reference Documents

The latest versions of the following documents apply to the development of a CDS:

ARINC Specification 429: *Mark 33 Digital Information Transfer System (DITS)*

ARINC Specification 664: *Aircraft Data Network*

1.9 Applicability

The CDS architecture should be robust with sufficient integrity, availability, reliability, and capacity to support any or all of the display types listed below. Also, it should enable growth to support other features that may be required in the future, as constrained only by what will physically fit in the cockpit. This CDS is not intended for cabin use.

Display types include, but are not limited to:

- Primary Flight Display (PFD)
- Navigation Display (ND)
- Head-Up Display (HUD)
- Multi-Purpose Control Display Unit (MCDU)
- Engine Indication and Crew Alerting System (EICAS)
- Multi-Function Display (MFD)
- Side Displays
- Data Link Control Display Unit (DCDU)

2.1 Introduction

This section describes the concept of operation for the standard protocol used between avionic equipment User Applications (UA) and the Cockpit Display System (CDS). This approach segregates the interactive event management and rendering details from the functional context displayed. The interface defined in this standard relies on a basic set of graphical user interface objects, hereafter referred to as "widgets."

The list of widgets is referred to as the ARINC 661 Human Machine Interface (HMI) Widget Library. It is described in detail in Section 3.2, HMI Widget Library. In general, these widgets correspond to a displayable entity. Some of these widgets are "interactive widgets" because they support crew member interaction using cursor control devices and keyboards. Crew member actions on interactive widgets are generally associated with event reports sent to the UA. The non-interactive widgets do not have any associated event.

This Specification defines a list of standardized widgets. CDS providers should include the widget library defined by this Specification in their display products. This is the interface between UA and CDS, and describes the widget interface to the UA, i.e., and widget parameters accessible to the UA.

The CDS should manage the actual rendering of the widgets as well as monitoring the flight deck crew interaction via display system input devices.

The UA should specify through the Definition File (DF), the characteristics of all the instances of **each widget it uses in the initial design or expects to use.** This is described in detail in Section 4.1.1, Definition File and User Application Layer Definition (UALD). These widgets are allocated inside the CDS.

The UA addresses **its** widgets through a run-time protocol. This is described in detail in Section 4.2, Run-time Communications. The UA animates the display format by setting the accessible parameters of the widgets, in order to reflect its functional context. The run-time protocol serves the purpose of the CDS reporting the crew events to the UA.

Characteristics and capabilities are the focus of this document, not the implementation of these capabilities.

2.2 Overview of Interface Level Between UA and CDS

The general approach for the widget interface is to segregate the UA functional description from the "look and feel" of HMI pages. The "look and feel" description refers to graphical characteristics such as color and border properties.

UAs should manage the widget interface in order to illustrate their functional state. Consequently, they should only manage functional states of widgets. UAs have no need to directly interact with "look and feel" characteristics.

The "look and feel" characteristics of a widget are linked with its functional characteristics. Thus, UAs may have to use a reference to a set of "look and feel" references in order to reflect their functional context. This service is provided by the widget parameter, "StyleSet." Refer to Section 3.1.3, Commonly Used Parameters, which defines a set of characteristics that should be defined by the airframe

manufacturer and stored inside the CDS. The airframe manufacturer specification task consists of defining the widget behavior implemented in the CDS, as well as the graphical characteristics associated with each particular functional state of the widget. UAs should refer to these pre-defined characteristics to manage the “look and feel” of their images, according to the airframe manufacturer-specified HMI rules.

This approach provides segregation between functional behavior managed by UAs and graphical behavior managed by the CDS. This provides a common look across all aircraft, and common implementation of behavior consistent with that airframe manufacturer’s cockpit philosophy. The style guide defined by the airframe manufacturer describes the “look and feel” inside the cockpit, and thus, provides the necessary information to UAs for their HMI interface design.

- There are two categories of widget interface definition, (1) specification or compile-time information and (2) run-time interface, described as follows: the compile time definition is static information stored inside the CDS that sets some parameters of the widget. The main objectives of this phase are to allow deterministic widget allocation in CDS memory, avoid heap memory utilization, and reduce system bus bandwidth requirements.
- Run-time interfaces enable UAs to control and change certain characteristics of the widgets during operation.

Figure 2.1-1 illustrates ARINC 661 protocol principles applied in a typical CDS system architecture. Higher values in the stacks take precedence over the lower modifiable values.

2.2.1 Definition Phase

The definition phase consists of loading and interpreting Definition Files (DF) in the CDS.

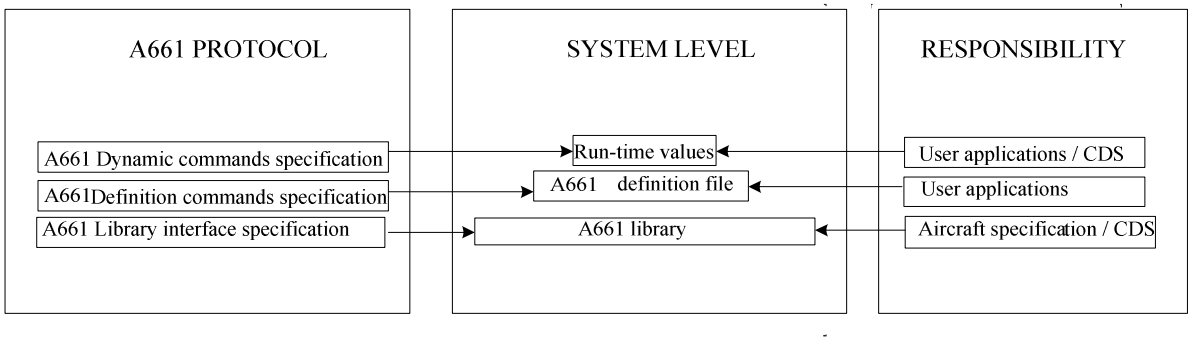


Figure 2.1-1 – ARINC 661 Protocol Principles

A DF is a loadable standard format file inside the CDS.

The DF specifies the creation of widgets that describe User Application (UA) interface pages.

The DF describes widget hierarchical structures.

The interpretation of the DF by the CDS results in the creation (instantiation and first setting of all parameters) of widgets.

All widgets should be created at this definition time to enable deterministic allocation of memory. The necessary memory size should be reserved at definition time for the allocation of the widgets. Items should be specified at run-time inside their container widget. Refer to Section 3.2.8, Map Management.

Some parameters can only be set at definition time. Among these parameters are all the parameters which have an impact on the memory size allocation.

The definition phase should be closed for a DF before the beginning of the **run-time phase for that DF (i.e., the** run-time data exchange for widgets defined inside the DF).

COMMENTARY

Defining the end of the definition phase and the beginning of the run-time phase is beyond the scope of this document. It is CDS integrator's choice to implement one global definition phase or an individual definition phase for each DF.

2.2.2 Run-Time Phase

The run-time phase consists of dynamic data transfers between UA and CDS using ARINC 661 run-time commands. These exchanges cover the following needs:

From UA to CDS:

- Updat**ing** the run-time widget parameters.
- Requests to the CDS for changes on entities managed by the CDS, for example layer visibility and direct focus motion.

From CDS to UA:

- Notification of event occurrence for application event processing.
- CDS configuration commands, for example, notification of application layer activation.

2.2.3 Special Conditions

2.2.3.1 Initialization

The CDS is the master of display configurations. The CDS determines the formats to be displayed and the UA Layers that will appear. Therefore, a UA should not transmit any data to the CDS before the CDS has notified the UA that its layer is ACTIVE (refer to Section 4 for such notification format).

After receiving such a notification, it becomes the UA's responsibility to update, as necessary, the parameters of the corresponding layer widgets AND to request the visibility of the layer. The CDS will not display the layer before this request is received in a message block.

2.2.3.2 Need for Re-initialization

In some conditions, the CDS may lose its image of the widget parameters as the UA has set them. In this event, the CDS can transmit a request for Layer Re-initialization. The UA should then update, as necessary, the layer widget parameters AND request the visibility of the layer. The CDS will not display the layer before this request is received in a message block.

2.2.3.3 Suppression of a Layer from Display

External conditions may lead the CDS to remove a layer from the display. In such a case, the CDS may notify the UA that the layer is INACTIVE. Upon such a notification, the UA should stop its update of the layer widget parameters.

2.2.4 ARINC 661 Conformance

A CDS conforms to ARINC 661 when: (a) it implements the prescribed CDS-UA communication mechanism (Section 4) including DF files and run-time messaging model, and (b) all widgets and widget parameters/capabilities that the CDS does implement are compliant with this standard.

An ARINC 661 CDS may implement some, but not necessarily all widgets defined herein. If the full possible range of widget attributes or behaviors is not supported, the CDS must at least behave in a robust and graceful manner with respect to unsupported requests by UA.

It is specifically counter to ARINC 661 to use non-standard widgets that are similar to standard widgets defined herein, without a justified change or improvement in capability. Those who wish to propose a new ARINC 661 widget are directed to Appendix G, which contains guidelines for addition of new widgets to the standard.

2.2.5 ARINC 661 Library Evolution

The ARINC 661 library should evolve in a manner that is compatible with the library defined herein. For example, if optional parameters were to be added to an existing widget, default values should be defined such that existing equipment can continue to use the older data block format. A new WidgetType ID should be created, defined, and used to indicate that the new size and format definition parameter block is in use.

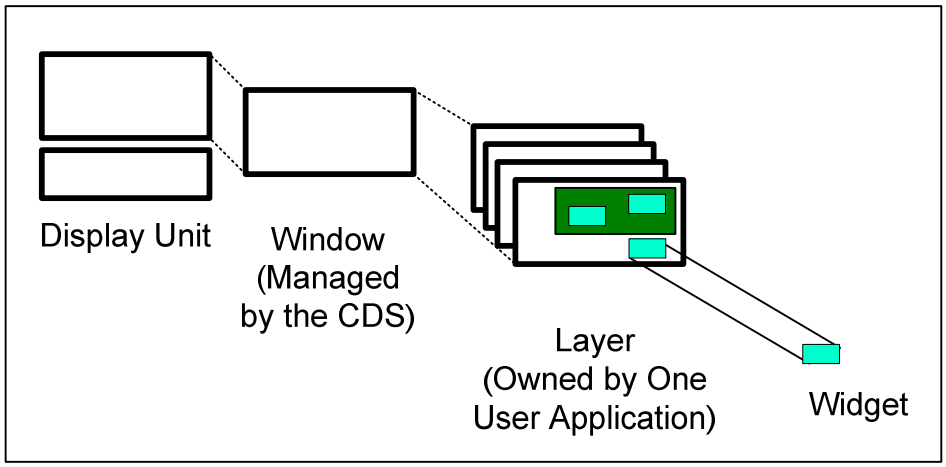


Figure 2.3-1 – Window/Layer Illustration

2.3 Window/Layer and General Concepts

This Specification uses a windowing concept, which can be compared to a desktop computer system windowing, but with many restrictions due to the aircraft environment constraints. Each format on a Display Unit (DU), shown in Figure 2.3-1, consists in a set of windows, defined by the current configuration of the CDS. A window is subdivided in layers. These layers are connected to the user applications and provide an area to display their widgets.

2.3.1 Window Definition

Windows are owned and managed by the CDS. In particular the CDS manages the visibility of the window. The UA may have no knowledge of the window set-up. Therefore, this section is provided as guidance and not considered a requirement for the interface between UA and CDS. Windows have the following characteristics:

- A format image rendered to a display unit surface is constructed from one or more windows.
- The windows included in a format image of a DU are fully defined in the configuration definition. The window visibility is managed by the CDS according to the current configuration.
- A window defines a rectangular physical area of the display surface.
- A window may not be resized.
- Windows cannot overlap each other.
- A window consists of one or more layers.

2.3.2 Layer Definition

A layer is the highest level entity of the CDS that is known by the UA. From the UA point of view, the Layer is the highest level container in the hierarchical structure of the UA widgets. From the CDS point of view, the layer is a graphical layer associated with an application inside a window. The definition of layer layout within a window is beyond the scope of this standard.

Layers provide the mechanism to combine information from several UAs inside one window.

COMMENTARY

Within an aircraft system, there is a need to place information from multiple client systems as well as information from the CDS itself within a single window. For example, the navigation display may require:

- Graphical information such as the compass rose
- Control widgets such as a PopUpMenu for changing the range
- Flight Management (FM) map
- TCAS information

2.3.2.1 Layer Graphical Definition

An ARINC 661 layer graphical definition has the following characteristics:

- It is a graphical layer inside a window
- A layer has an origin that is defined with respect to the origin of its window. For a special case, refer to Section 3.3.7, Connector
- All rendering within a layer is clipped by the bounding window definition
- Layers may overlap

In the hierarchical structure of the DU format image definition, the layers are containers just under the window level.

2.3.2.2 Layer Content Management

Layer content management has the following characteristics:

An ARINC 661 Layer is associated with a UALD. Thus, each layer has only one owner, that is, the owner of its associated UALD. A layer contains the hierarchical structure of widgets defined inside its associated UALD.

COMMENTARY

A layer can be displayed in several windows at the same time. If the duplicated layer is interactive, it could lead to interactive widget identification confusion.

One proposal could be to allow the interactivity by the CDS only on one of these layers.

The UA, as well as the CDS itself, may be an owner of a layer.

The UA has only knowledge of its layer, **and does** not **have** knowledge of the containing window, which is defined by current CDS configuration.

A UA or the CDS itself, possibly, **may** own several layers within a window.

The owner of a layer is responsible for managing the parameters of **all widgets** contained **in that layer**. The owner UA should know the complete set of run-time parameters for widgets contained inside the layer.

A crew-member input through an interactive widget contained within a layer transmits an event to the owning UA.

2.3.2.3 Layer Priority Management

Layer content management has the following characteristics:

- Layers are assigned a static priority that defines the order of visibility, e.g., which layer appears on top of other layers. A UA knows the relative priority of its own layers, while the CDS manages absolute priority between layers of different UAs. Priority between layers of different UAs is not accessible to UAs.
- Widgets are drawn in the order they are defined in the UALD, so that the last defined is drawn on top of the others. Note that if container C1 is defined before container C2, then all widgets included in C2 will be drawn on top of all widgets defined in C1.

Some widgets should always be drawn on top of the other widgets, for example, ComboBox, PopUp Menu.

2.3.2.4 Layer Activity/Visibility Management

A layer has two properties: Active/Inactive and Visible/ Invisible. For definition of commands to manage these properties, refer to Section 4.4.3.2, Request/Notification from CDS to UA.

2.3.2.4.1 Visibility

The layer visibility is managed by the UA through a visibility parameter.

All objects within a layer should become invisible when the layer becomes invisible by control of the layer visibility parameter. This does not affect the current value of each widget visibility parameter.

2.3.2.4.2 Activity

The activity of the layer is controlled by the CDS. When a layer is active, the CDS should update the data from the UA that owns the layer, even if the layer is not visible. Refer to Section 4.4.3, ARINC 661 Request/Notification for A661_REQ_LAYER_ACTIVE.

The CDS sends the A661_NOTE_LAYER_IS_ACTIVE request to the UA when CDS activates the layer.

The CDS sends a A661_NOTE_LAYER_IS_INACTIVE request to the UA when CDS de-activates the layer.

When a layer becomes inactive, its visibility is turned off by the CDS. When the layer becomes active, it is the responsibility of the UA to turn on the visibility of its layer. Also, when a layer becomes active, the UA should reinitialize the layer's data.

COMMENTARY

The specific use of the Activity / Inactivity property on a layer of the CDS is outside the scope of this standard. This should be defined by the airframe manufacturer, except at initialization, the CDS should send A661_NOTE_LAYER_IS_ACTIVE notification.

When the layer is inactive, the CDS has only to consider the A661_REQ_LAYER_ACTIVE request command from the UA owning the layer.

2.3.2.5 Layer Context Management

Context management covers the notion of correlation between the data exchanged and the data displayed at a given time.

A “context number” is attached to each layer. The value management of this parameter is the responsibility of the UA owning the layer. The application will modify the context number through the context number parameter of the block structure.

The CDS sends the current context number of the layer containing the interacted widget inside the block structure.

The initial value of the context number is set inside the layer definition block (UALD).

Context number allows the UA to manage the internal state of its display in the CDS.

2.3.3 Configuration Issues

The CDS controls the configuration of the cockpit by defining the following:

- The correct window to go on the specific DU.
- The correct application layer to go in the specific window. The CDS notifies the UA that a window containing layers of this UA is displayed and the UA has to be ready for widget management through notification A661_NOTE_LAYER_IS_ACTIVE.

A UA can send the CDS a request to display one of its layers. The CDS may accept or reject this request depending on the configuration logic that is implemented at that time. Refer to Section 4.4.3, ARINC 661 Request/Notification for A661_REQ_LAYER_ACTIVE.

2.3.4 Positioning and Size Within Window

Figure 2.3.4 illustrates graphic references for widget positioning.

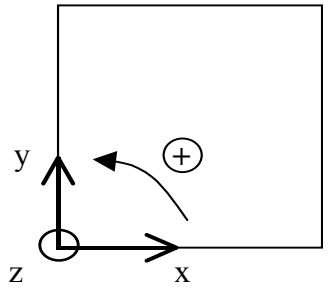


Figure 2.3.4 – Graphic references for widget positioning

2.3.4.1 Origins

All origins are in the lower left-hand corner of the object. Origin of widgets within containers is relative to the immediate container.

Any exception to these provisions will be clearly stated in the detailed description of the widgets.

2.3.4.2 Angles

All angles are measured in degrees. All rotation is around the Z-axis. The ZERO degree is along the X-axis in the positive direction. The positive direction of rotation is in the counter-clockwise direction from the X-axis. When specifying an arc, the arc becomes a complete circle when the StartAngle and EndAngle represent the minimum and maximum possible values of fr(180).

Any exception to these provisions will be clearly stated in the detailed description of the widgets.

2.3.4.3 Screen Units of Measurements

All screen units should be measured in units of millimeters with a resolution of 0.01 millimeters. Therefore, the position and size of widgets are expressed with a resolution of 0.01 millimeters. Widget position parameters are signed integer and widget size parameters are unsigned integer. Refer to Section 3.2, HMI Widget Library Summary. Any exception to these provisions is clearly stated in the detailed description of the widgets.

2.3.5 Cursor Management

The cursor is controlled by the CDS. The cursor shape is defined by the CDS. The cursor shape is an element of the “look and feel” of the cockpit and is managed in a homogeneous way throughout the different formats. The cursor shape depends on the type of the widget that holds the cursor (e.g., button, text editor) and the current state of this widget (e.g., editing mode). Nevertheless, some information related to the cursor may be exchanged between the CDS and the UAs.

Two key terms are most important to the understanding of the cursor management in ARINC 661. They are respectively: Focus and Highlight.

- An interactive widget is focused when it receives the events triggered by a crew member through non-CCD input devices (such as keyboard).
- An interactive widget is highlighted when the cursor passes over its interactive area. Depending on the implementation, the click may select and/or focus the widget.

Focus and highlight are defined as two independent characteristics of an object; however, the relationship between them is implementation dependent. Focus and Highlight may change the graphical look of a widget.

2.3.5.1 From UA to CDS

An example of cursor management from the UA to the CDS is to request the **focus be put on** a particular widget. This request may or may not move the cursor according to the CDS implementation. Refer to Section 4.4.3.1, Request from UA to CDS.

Another example is the ability to inform the CDS at definition time of widget navigation order, which supports CDS management of focus navigation. Refer to Section 3.1.3.5, Parameters Relative to Focus Navigation.

2.3.5.2 From CDS to UA

An example of cursor management from CDS to UA is the identification of which cursor, i.e., pilot or co-pilot, has been used to interact with a widget in addition to the other information related to the event.

COMMENTARY

Some cursor characteristics are outside the scope of this Specification. They should be defined by the airframe manufacturer for the display system provider including the following features:

- Interactivity features
- Movement rules between DUs
- Movement rules between windows
- Link between the cursor shape and the window characteristics, for example frozen window
- **Response to cursor events involving overlapping widgets**

Determination of all cases where cursor snaps might occur. For example, when the cursor is on a widget owned by a UA that suddenly fails, the placement of the cursor should be defined. The cursor could go to another widget in such a case. Another case to consider is where to put the cursor when a window or layer is first initialized and displayed. The definition and use of default locations for such cases should be considered.

Precise response time requirements depend on user system operational requirements. Table 2.3.5.2 provides guidelines that should be considered by system designers in determining computer processing requirements and software architecture necessary to support this interface.

The CDS provides the ability to perform the first four of these tasks within itself, drastically reducing the processing load on the user system, if used properly.

Table 2.3.5.2 – Guidelines for Cursor-Control Timing

Task Description	Time
Time between cursor collision with display object and indication of collision (cursor shape change or object highlight)	50 ms max
Time between object selection and indication of selection.	180 ms max 150 ms avg
Time between crew movement of the CCD and cursor movement on the display.	100 ms max 80 ms avg
Time between cursor command for paging and menu selection and resulting user system display.	300 ms max
Time between cursor command action and resulting action of the command being processed (for commands other than paging or menu selection).	1 s max

COMMENTARY

System integrators and designers are reminded that the flight deck is not an office desk-top environment. Thus, common desk-top practices such as “double click” may be difficult to implement successfully. Turbulence may produce an unintended double click. Data transmission rates may make it difficult for the display system to recognize a double click. Therefore, if a double click feature is used in the system design, the CCD should bear the responsibility for recognizing this situation and transmit it as a discrete event to the display system.

COMMENTARY

In cases where the active areas of two interactive widgets overlap, if the pilot makes a selection in the overlap area, it is recommended the CDS send the applicable event associated with the widget on top. A widget is defined to be on top of another widget if the widget is listed after the other widget in the DF and shares at least a portion of the display space with the other widget. This may not be applicable to all widget types (e.g., CursorOver).

In cases where the active areas of one or more interactive MapItem or MapSource widgets overlap, the sending of one or more events will be CDS dependent.

3.1 Introduction to Widgets

Communication between the CDS and UA is defined based on the identification of widgets defined in this section, Widget Library.

3.1.1 Widget Identification

A widget is defined with respect to the UA to which it belongs. Widget identifiers are assigned and managed by the UA. A widget identifier, referred to as [WidgetIdent], is unique in one User Application Layer Definition (UALD).

Since the CDS manages layers and their priorities, the CDS needs to know at definition time to which layer a widget belongs. Therefore, the CDS also needs a relative [LayerIdent] from the UA. A [LayerIdent] referenced by the “User Application A” could be identical to a [LayerIdent] referenced by the “User Application B.” Internally the CDS resolves its internal Layer Identification by using the [User Application Ident].

At definition time, the interface between CDS and UA should uniquely define widgets by the combination: [UserApplicationIdent].[LayerIdent].[WidgetIdent]

COMMENTARY

At run-time, [UserApplicationIdent] is resolved by the CDS using information from the system bus.

3.1.2 Widget States

3.1.2.1 Widget States Definition

Four different levels, illustrated in Figure 3.1.2 define widget states:

- Visibility level: widget is visible or not
- Inner level: specific states of a widget. This level represents the core of the widget behavior as well as its functional objectives. Examples of inner states:
  - for a basic PushButton, there is one stable inner state
  - for a CheckButton, there are two stable inner states, which are “selected” and ”unselected”
- Ability level: widget is enabled or disabled. This level exists for interactive widgets. An enabled widget is ready to receive input from crew member interaction
- Visual level (visual representation): internal behavior of the widget inside the CDS. Examples of visual representation are Normal and Focus. Refer to the glossary in Appendix A for the definitions of the visual states listed below.

State levels 1, 2 and 3 describe the possible combinations of states accessible to a UA in order to interact with a widget. These states affect the behavior of the widget. These widget states can be managed through run-time parameters, specifically:

- Visible
- Specific parameter related to the inner states (like “CheckButtonStates” for a CheckButton)
- Enable

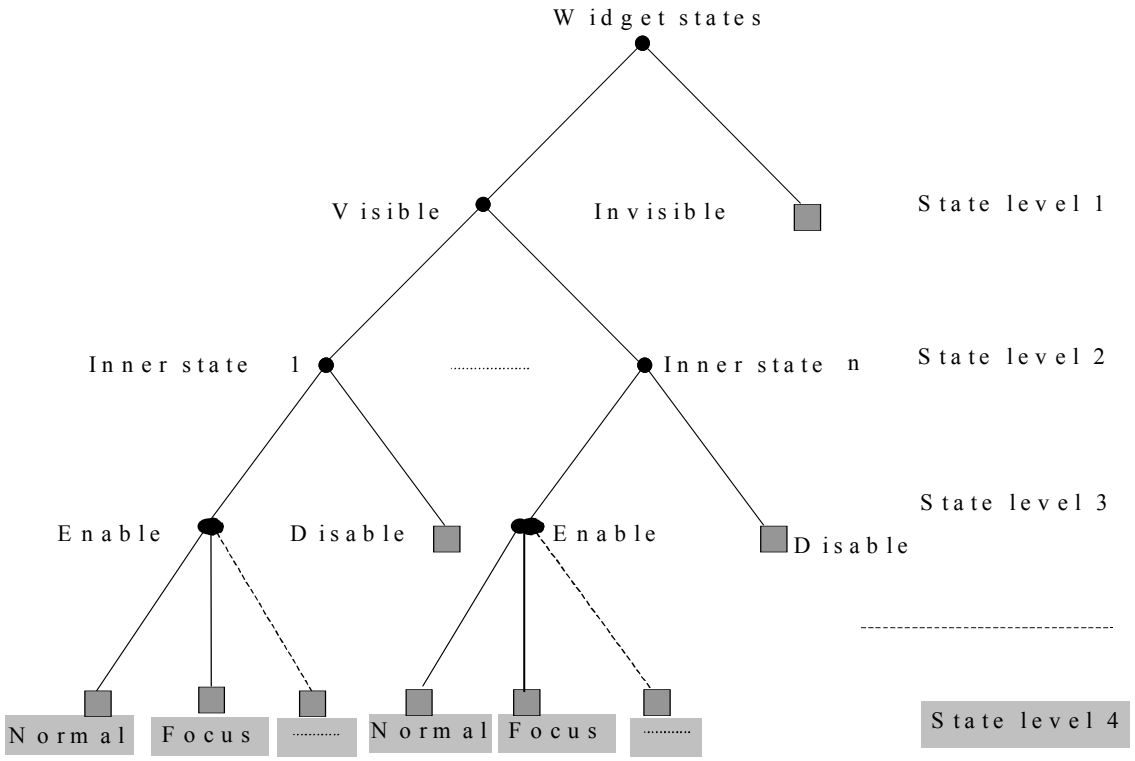


Figure 3.1.2 – Example of Widget States Levels

State level 4 is the visual representation. In the ARINC 661 CDS interface, the complete definition of the visual representations might freeze the widget behavior internal to the CDS. To avoid this, visual representation should be part of the aircraft original equipment manufacturers (OEM) specification and implemented by the CDS supplier in the CDS Widget Library.

A UA should not have any direct access to the visual representations. Therefore, visual presentations do not have to be defined within the ARINC 661 interface protocol. Only the ARINC 661 parameter effects on graphical representation should be described in the ARINC 661 interface. The style guide defined by the OEM should describe the “look and feel” and thus, provide necessary information to UAs for their HMI interface design.

3.1.2.2 Inner State Management: "Race Condition"

Both CDS and the owner UA of a widget can manage the inner states of a widget. For instance, considering a CheckButton:

a. **Upon selection by a crew member, the CDS will change the widget inner state from SELECTED to UNSELECTED or from UNSELECTED to SELECTED.**
b. **The UA may change the inner state to initialize or refresh the interface.**

When the CDS changes the state of a widget based upon crew member interaction, it sends an event to inform the UA of the interaction. In this case, the widget is considered as an IN widget.

If the Widget is non-interactive (e.g. CheckBox with Enable=F), then the inner state is only controlled by the UA and the widget is an OUT widget.

But the inner state may represent an actual system state of the UA and the UA will need to update the inner state based upon its knowledge of the state as well as accept modification from the pilot. For example, a CheckButton that is selectable by either the Capt or FO. In this case, when either pilot changes the inner state, the UA will need to update the state of that widget on the off-side display.

In the case of an IN/OUT widget, a "race condition" can occur that may need to be addressed by the CDS and/or UA. The issue that can occur is that shortly after the CDS sends a notify widget event to the UA that an inner state has changed, the CDS may receive from the UA a set parameter command to change the inner state which was sent prior to the UA receiving the notify event. In this case the CDS does not know if the set parameter command was sent prior or after the UA received the notify widget event and the commanded state may not be consistent with the newly updated widget state. The result may be a momentary reversion of the displayed inner state after the widget is selected.

This same condition can also occur with "entry" widgets such as EditBoxText. In general, EditBoxText widgets would be IN/OUT since the format of the entry into the widget is often different than the format of the displayed entry. For example, if the EditBoxText contains weight represented as "XXX.X", and the pilot wishes to enter a new value, the allowed entry may be "XX" which the UA would then convert to " XX.X" and update the content of the EditBoxText. In this case, the race condition would result in the widget reverting back to the XXX.X value prior to the pilot entry, until the UA response with the updated entry value.

Widgets which potentially have his issue include: CheckButton, ComboBox, EditBoxMasked, EditBoxNumeric, EditBoxText, PictureToggleButton, ToggleButton, ComboBoxEdit, EditBoxMultiline.

A number of approaches can be taken by the UA and/or CDS to address this issue. The following approaches are possible methods that could be used by a system design. The specific approach used (one of these or others) used by a specific system design is outside the scope of this specification.

- **The race condition can be mitigated to some degree by minimizing the update of IN/OUT widget by the UA. Care must be taken to insure that the CDS inner state of the widget is consistent with the UA state of the widget.**
- **For "EditBox" Widgets, the race could also be minimized by the UA using the EditBox_Open and EditBox_Close event to restrict sending A661_String to the Widget while it is actively being edited.**
- **For those widgets with an inner state, the CDS implementation could delay application of received parameters (subject to a timeout) to the Widget for which it has sent an A661_EVT_STATE_CHANGE until it receives a Set Parameter command from the UA which reflects the notified inner state.**
- **The CDS implementation could also mitigate the race condition by applying a "timeout" period after sending a notify event to the UA during which it will delay application of Set Parameter messages to the widget until the timeout expires. The UA is then required to respond to the notify event with updated Parameters prior to the timeout expiring.**

3.1.3 Commonly Used Parameters

This section includes tables that identify the parameters commonly used by all widgets of the ARINC 661 library.

3.1.3.1 Identification of the Widget

Widget Identification Parameters are defined in Table 3.1.3.1.

Table 3.1.3.1 – Widget Identification Parameters

Parameter	Description
WidgetType	Type of widget
WidgetIdent	Identifier of the widget (refer to Section 3.1.1,) WidgetIdent is a non-null positive value ([WidgetIdent] >0). NULL is reserved for referring to the layer level (e.g., ParentIdent)
ParentIdent	Identifier of the immediate container of the widget. Only a special category of widgets called "Container" can be the parent of other widgets. At the highest level of the widget hierarchy within a layer, the ParentIdent value is 0 (NULL). This means that the parent of the widget is the layer.

3.1.3.2 States of a Widget

Widget State Parameters are defined in Table 3.1.3.2.

Table 3.1.3.2 – Widget States Parameters

Parameter	Description
InnerState	Holds the specific functional state (if any) of a widget. The set of possible values depends on widget type.
Visible	A661_FALSE: The widget will not be rendered. A661_TRUE: • If all its parents are visible, the widget will be rendered. • If any one of its parents are invisible, the widget will not be rendered, whatever the value of its visible parameter.
Enable	A661_FALSE: The widget will not be interactive. A661_TRUE or A661_TRUE_WITH_VALIDATION: • If all its parents are enabled, the widget will be interactive. • If any one of its parents are disabled, the widget will not be interactive, whatever the value of its Enable parameter. An invisible widget is not interactive, independent of the value of its Enable parameter.
Anonymous	A661_FALSE: run-time accessible. Widget can be modified at run-time, if it has some run-time accessible parameters. A661_TRUE: anonymous. Widget can not be modified at run-time by UA. CDS behavior when a UA attempts to SetParameter on an anonymous widget is undefined.

3.1.3.3 Look and Feel Characteristics of a Widget : "StyleSet" Parameter

Widget State Parameters are defined in Table 3.1.3.3.

Table 3.1.3.3 – Widget StyleSet Parameter

Parameter	Description
StyleSet	StyleSet allows the UA to select from a predefined set of graphical characteristics to be applied to a widget. This serves two purposes. First, many graphical capabilities (color depth, halo, fill styles, line weights/patterns, blinking, transparency, fonts, character highlighting, kerning, rotation, etc.) are inherently a function of CDS architecture. Guidance requiring or disallowing any of these characteristics is beyond the scope of this document. Second, the application of these characteristics is usually intended by the aircraft OEM to be consistent across all UAs for common state conditions. Indexing among predefined styles supports this goal. Common state conditions can be defined as conditions that impact more than one user application in the same way (e.g., Alert, Caution). It can also be used by a single application for convenience and control of hidden characteristics. Thus, any graphical characteristics set by StyleSet that match individually accessible graphical characteristics will be overridden by the values specified

Parameter	Description	
	in the StyleSet. All other parameters take on their default values. Hidden graphical characteristics used for representing common state conditions are only accessible via StyleSet commands. This Specification defines one default SyleSet value: STYLE_SET_DEFAULT meaning that default graphical characteristics will be used. Aircraft OEM (or CDS supplier) defines the list of StyleSet values. Examples of possible StyleSet values:	
	STYLE_SET_NOMINAL	STYLE_SET_SELECTED
	STYLE_SET_ADVISORY	STYLE_SET_PRESELECTED
	STYLE_SET_CAUTION	STYLE_SET_ENGAGED
	STYLE_SET_WARNING	STYLE_SET_ARMED
		STYLE_SET_NOT_ENGAGED

3.1.3.4 Positioning/Size of a Widget

Widget Position/Size Parameters are defined in Table 3.1.3.4.

Table 3.1.3.4 – Widget Position/Size Parameters

Parameter	Description
PosX	The X position of the widget reference point is an offset with respect to the absolute X position of the reference point of the widget container (parent).
PosY	The Y position of the widget reference point is an offset with respect to the absolute Y position of the reference point of the widget container (parent).
SizeX	The X dimension size (width) of the widget.
SizeY	The Y dimension size (height) of the widget.

The PosX, PosY, SizeX and SizeY parameters define a clipping area for the widget. Graphical characteristics must not be rendered outside this area.

This area defines the static area of the widget. For widgets containing a “Pop Up part” such as ComboBox, only the static part is constrained by these parameters.

3.1.3.5 Parameters Related to Focus Navigation

Management of directional motion of the focus, e.g., through arrow keys, is internal to CDS, and therefore does not require interface level parameters.

However, it is possible for the UA to specify a “logical” navigation order through the use of a given key (for example, tabulation). This can be done using the “NextFocusedWidget” parameter.

To allow automatic motion of the focus after a selection or a confirmation event, a Boolean parameter “AutomaticFocusMotion” will be used in combination with the “NextFocusedWidget” parameter.

Widget Common Structure is defined in Table 3.1.3.5.

Table 3.1.3.5 – Widget Common Structure

Parameter	Description
NextFocusedWidget	Widget ident of next widget to be focused upon crew member validation.
AutomaticFocusMotion	A661_FALSE: No automatic motion: after a crew member validation, the focus remains on the widget until an explicit move of the focus. A661_TRUE: Move automatically the focus after a crew member validation to the next widget according to the NextFocusedWidget parameter

Focus can be moved among widgets residing in a layer, but can also be moved among widgets residing in different layers, even if those layers are owned by different User Applications. Focus navigation between widgets in different layers is discussed in Section 3.6.4.

3.1.4 Widget Events

Widgets notify User Applications of events caused by crew actions. More precisely, when the human operator uses the CDS to act on an interactive widget, and the action generates an event, the CDS notifies the UA owning the layer that contains the widget, according to the structure defined in Table 4.5.4.2-3.

The events that each widget can generate are listed in tables that appear with the widget definition, generally shown between the Creation Structure Table and the Runtime Modifiable Parameters Table for that widget.

Widget events (A661_NOTIFY_WIDGET_EVENT) are the result of human actions. Events are not the result of a CDS-UA interaction where the resulting event is either redundant, or represents a state change that can be easily determined by the UA.

An example of events resulting from human actions: assume a RadioBox containing two ToggleButtons (and/or CheckButtons), one of them currently selected. If a person selects the other button, the newly selected button sends an A661_EVT_STATE_CHANGE event with the state field set to A661_SELECTED. The CDS is required to set the other button to the unselected state, so that button does not send an event to the UA. Sending an “unselected” event is redundant (wastes resources), and dealing with it may complicate the UA by forcing it to discern the difference between an required CDS behavior and a race condition (see Section 3.1.2.2).

An example of a redundant event: if a UA uses A661_CMD_SET_PARAMETER to deselect a button in the previous example, or to put a different button in the selected state, no A661_NOTIFY_WIDGET_EVENT(s) would be generated, for the reasons stated above. It may seem desirable in some cases to have an “acknowledge” of the command, but most SetParameter commands do not accommodate such an acknowledge. Some solution to that problem (which might be a lossless communication path or a lower-level communication protocol) must exist, and that solution makes this “acknowledge” event redundant.

Widgets that trigger events are listed in Table 3.1.4.

Table 3.1.4 – Widget Event Cross Reference

Widgets \ Events	A661_EVT_CURSOR_ENTER	A661_EVT_CURSOR_INSIDE	A661_EVT_CURSOR_EXIT	A661_EVT_CURSOR_POS_CHANGE	A661_EVT_EDITBOX_OPENED	A661_EVT_FIRST_VIS_ENTRY_CHANGE	A661_EVT_FRAME_POS_CHANGE	A661_EVT_INCREMENT	A661_EVT_ITEM_SYNCHRONIZATION	A661_EVT_POPUP_CLOSED	A661_EVT_POPUP_PANEL_CLOSED	A661_EVT_SEL_ENTRY_CHANGE	A661_EVT_SELECTION	A661_EVT_SELECTION_MAP	A661_EVT_STATE_CHANGE	A661_EVT_STRING_CHANGE	A661_EVT_STRING_CHANGE_ABORTED	A661_EVT_STRING_CONFIRMED	A661_EVT_TABBED_PANEL_CHANGE	A661_EVT_VALUE_CHANGE	A661_EVT_WATCHDOG_EXPIRED	A661_EVT_WATCHDOG_NORMAL
ActiveArea													X									
CheckButton															O							
ComboBox												X										
ComboBoxEdit					X							X				X	X	X				
CursorOver	X	X	X																			
CursorPosOverlay				X																		
EditBoxMasked					X											X	X	X				
EditBoxMultiLine					X											X	X	X				
EditBoxNumeric					X											X	X	X				
EditBoxNumericBCD					X											X	X	X				
EditBoxText					X											X	X	X				
MapHorz									X													
MapHorz_ItemList													X									
MapHorz_Source								X						X								
MapVert									X													
MapVert_ItemList													X									
MapVert_Source														X								
PicturePushButton													X									
PictureToggleButton															X							
PopUpMenu										X												
PopUpMenuButton										X												
PopUpPanel											X											
ProxyButton													X									
PushButton													X									
ScrollList						X						X										
ScrollPanel							X															
SelectionListButton												X										
Slider																				X		
TabbedPanelGroup																			X			
ToggleButton															X							
WatchdogContainer																					X	X

O – Event has the same name as used in other widgets but contains different values.

3.2 HMI Widget Library Summary

3.2.1 Widget Summary

Table 3.2.1 summarizes the Widget Library.

Table 3.2.1 – Widget Library Summary

Widget Type	Description
ActiveArea	The Active Area widget is a transparent rectangular area defining an interactive area. Selection of this area will send an event to the owner application.
BasicContainer	The BasicContainer widget manages the visibility and the interactivity of a group of widgets.
BlinkingContainer	The purpose of the BlinkingContainer widget is to apply blinking behavior to a group of widgets.
BufferFormat	This widget provides a means for compressing data from different widgets (in the same layer) in one buffer. Use for this widget could be initialization of a page, or refresh of big widget. The bufferFormat format is defined at definition phase. The content of the buffer is exchanged at run-time from the UA to the CDS.
CheckButton	A CheckButton allows the crew-member to select or not an option. (other names: Radio/Toggle box).
ComboBox	A ComboBox is a widget providing a means to select one item among a list. This widget is composed of a static part displaying the selected item and a pop up part displaying the ScrollList of items.
Connector	The purpose of this widget is to connect a layer to a container of another layer. In this way it provides the means for a master application to interact on widgets owned by another UA. Typical use cases are for TabbedPanelGroup, MapHorz which can mix data from several user applications.
CursorPosOverlay	A CursorPosOverlay consists of a transparent rectangular area of the display. The distinguishing characteristic of a CursorPos Overlay is that the reportable event is the current cursor pointer position relative to the CursorPosOverlay position.
EditBoxMasked	The Masked edit box is an extension of the Text edit box. The difference with the basic Text edit box is that some characters are not modifiable by the crew member. Those Characters non-modifiable are specified by the user application by setting to 0 the “alpha mask” parameter and the “numeric mask.”
EditBoxNumeric	The numeric edit box allows editing a numeric value. A crew member can modify this value using its input devices. As it is a numeric value, CDS is able to increment itself the value. The widget can receive a number of increment or a numeric key value.
EditBoxText	A text edit box allows to display a string, which can be modified by the crew member (other names: Text field, Text entry box).
GpArcEllipse	The graphical primitive GpArcEllipse allows the definition of an arc (portion of an ellipse or a circle).
GpArcCircle	The graphical primitive GpArcCircle allows the definition of a circular arc.
GpCrown	The graphical primitive GpCrown allows the definition of a circular filled region.
GpLine	The graphical primitive GpLine allows the definition of a line.
GpLinePolar	The graphical primitive GpLinePolar allows the definition of a line using a polar definition.
GpRectangle	The graphical primitive GpRectangle allows the definition of a rectangle.
GpTriangle	The graphical primitive GpTriangle allows the definition of a triangle.
Picture	A picture is a reference to an image available in the CDS. The picture reference can be modified by the user application. Picture may have different

Widget Type	Description
	color not modifiable (unlike characters). Picture has no rotation capability.
Label	A Label consists of a non-editable text field at a defined display location.
LabelComplex	A Complex Label consists of a non-editable text field at a defined display location. The graphical representation is managed through an escape sequence.
MapHorz_ItemList	MapHorz_ItemList represents a group of related graphics. Example use of the widget is the creation of flight plan, map background symbols and TCAS intruders.
MapLegacy	Map Legacy widget provides a means for being compatible with currently use data format, such as ARINC 702 format for FMS, AEEC 453 format for weather radar.
MapHorz_Source	MapHorz_Source is a specialized container. It contains widgets expressed in a common coordinate system. It describes characteristics of the common coordinate system.
MapHorz	MapHorz consists of a rectangular region on the display, which contains reference information to allow the display of map features in the cockpit. It allows multiple sources of information with different coordinate systems to be fused into a composite map image.
MaskContainer	MaskContainer is intended to apply a referenced mask to a group of widgets.
Panel	A panel groups several widgets together in a rectangular area and has clipping capabilities.
PicturePushButton	Momentary switched button with Picture, which allows the crew-member to launch an action (to send an event to the owner user application).
PictureToggleButton	Two stable states button with Picture.
PopUpPanel	PopUpPanel is a container displayed on the top of other layers. PopUpPanel visibility can be managed by the CDS using logic defined by the OEM. Because PopUpPanel is displayed on the top, it should not be used as a regular container.
PopUpMenu	PopUpMenu is a set of selectable items. This menu is displayed on the top of other layer, but it is affected by clipping area of it parents. PopUpMenu visibility is managed by the CDS.
PopUpMenuButton	PopUpMenuButton is a button providing the ability to display a PopUpMenu. This mechanism is internal to the CDS.
PushButton	Momentary switched button, which allows the crew-member to launch an action (to send an event to the owner user application).
RadioBox	Manages the visibility and the interactivity of a group of CheckButtons or ToggleButtons. The selection of one of the CheckButtons or the ToggleButtons is exclusive.
RotationContainer	A RotationContainer has the same capabilities of a BasicContainer. It allows a rotation transformation to be applied to the children of the container.
ScrollPanel	A ScrollPanel is a “sheet container widget”, for which only a subpart is visible called the “frame”. Scroll controls provide the capability to scroll the visible part inside the whole sheet.
ScrollList	A ScrollList is a list of items, for which only a subpart is visible. Scroll controls provide the capability to scroll the visible part of items inside the whole list.
Symbol	Symbol has rotation and color capability. Symbol widget has a reference to a table.
TabbedPanel	A TabbedPanel widget is a Panel associated with a selection button. This widget is only for use within a TabbedPanelGroup widget.
TabbedPanelGroup	A TabbedPanelGroup groups several TabbedPanel widgets. A TabbedPanelGroup allows the user application or a crew member using a selection button to display one of the TabbedPanel widgets. All of the panels inside the TabbedPanel widgets occupy the same display space. Only one may be displayed at a time.

Widget Type	Description
ToggleButton	Two stable states button with text.
TranslationContainer	A TranslationContainer is similar to a BasicContainer. It allows a translation transformation to be applied to the children of the container.
WIDGET EXPANSION SUPPLEMENT 1	
ComboBoxEdit	Like ComboBox, ComboBoxEdit provides a means to select one item in a list of items. This widget is composed of a static part displaying the selected item and a pop up part displaying possible items.
EditBoxMultiLine	EditBoxMultiLine is a text edit box for displaying text across several lines in a scrolling area.
ExternalSource	The function of the ExternalSource widget is to specify to the CDS where an external input should appear on the display. For example, an external input may be a video signal input or a bitmap image.
MapGrid	MapGrid provides a means for conveying arrays of data to the CDS that are rendered as area fills. The intended use is for filling areas on background layers of the NAV window with colors and/or patterns that indicate terrain topography, precipitation intensity, or other irregular, dynamic data.
MapVert	The MapVert widget is the counterpart of the MapHorz widget for a vertical display made of a slice presentation. It is based on Cartesian coordinate system.
MapVert_ItemList	The MapVert_ItemList is equivalent to the MapHorz_ItemList for vertical displays. A MapVert_ItemList contains a list of Items to be drawn.
MapVert_Source	The MapVert_Source is the equivalent of the MapHorz_Source for vertical displays. The MapVert_Source widget is a specialized container. It contains some MapVert_ItemList widgets to display Items expressed in a common coordinate system.
MenuBar	A MenuBar is a widget containing PushButtons, PicturePushButtons and PopUpMenuButtons. It implements specific behaviors to move from one button to another.
WIDGET EXPANSION SUPPLEMENT 2	
MutuallyExclusive Container	The MutuallyExclusiveContainer groups children widgets and provides control to assure that only one child is visible at the same time. The MutuallyExclusiveContainer has no graphical representation.
ProxyButton	The ProxyButton directs select event from physical keys present in the CDS to any widget with a select event.
WatchdogContainer	This widget is used to assure certain sets of parameters are refreshed at a defined rate. If the timer is not refreshed at the required rate the CDS sends an event to the UA and sets automatically displays a predefined child widget.
Slider	A Slider allows the crewmember to select a value between the range of MIN_VALUE and MAX VALUE. The Slider can be displayed in either the horizontal or vertical axis.
PictureAnimated	A PictureAnimated is a reference to a set of images available in the CDS that can not be modified by the user application. By displaying this set of pictures successively at a frequency defined as a parameter of the widget, the CDS performs an animation.
SymbolAnimated	The SymbolAnimated, widget is defined by a sequence of symbol references, along with orientations and relative movements for each cycle of the animation.
SelectionListButton	The SelectionListButton allows a crew member to select one entry within a list. This widget is composed of a fixed label and a pop up part displaying the ScrollList of items.
WIDGET EXPANSION SUPPLEMENT 3	
EditBoxNumericBCD	**The EditBoxNumericBCD is very similar to EditBoxNumeric and has the same general features except that it allows the entry of non base 10 values such as latitude or time.**

Widget Type	Description
CursorRef	**This widget is used to define a screen or map location that can be used with the A661_REQ_CURSOR_ON_WIDGET command.**
CursorOver	**This widget is similar to the ActiveArea widget, but generates events as soon as the cursor enters the widgets active area.**
FocusLink	**This widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.**
FocusIn	**Together with FocusOut, this widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.**
FocusOut	**Together with FocusIn, this widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.**
SizeToFitContainer	**This widget is used to dynamically size a set of child widgets so that they are all the same size. The function can be applied in the vertical or horizontal axis.**
ShuffleToFit Container	**This widget is used to arrange a set of child widgets so that there is no unused space between them. If a child is made invisible all of the remaining children are moved to close up the space.**

3.2.2 Widget Classification

Table 3.2.2-1 describes the categories of widgets in the Widget Library. Table 3.2.2-2 defines the widget classifications. These categories are not exclusive, and a widget may belong to several categories.

Table 3.2.2-1 – Widget Library Categories

Widget Category	Description
Container	Container is a widget that can be referenced as a parent. A Container groups several widgets together. This category of widget is used to design the hierarchical structure of the widget inside the HMI pages.
Graphical Representation	Category of widgets which have a graphical representation.
Text string	Category of widget that displays a string of text.
Interactive	Category of widgets on which the crew member can interact. An Interactive widget has an Event Structure table attached (refer to Section 3.0, Widget Library).
Map management	Category of widgets related to the management of the Dynamic widget inside map. Typical use case for this symbol is Navigation Display format.
Dynamic motion	Category of widgets which can change of position at run-time.
Utility	**Category of widgets that are not containers, do not have graphical representation and that are not interactive. These widgets have specific functionality in order to extend or optimize the ARINC 661 defined principles.**
UA Validation	**Category of widgets which may have their events (if applicable) validated by the UA. The codes are defined as follows:** **A = Supports A661_ENABLE but UA Validation is not applicable for this widget type** **B = Supports A661_ENABLE and uses A661_ENTRY_VALID** **See Section 3.2.9 for more information.**

Table 3.2.2-2 – Widget Classification Table

	Widget Categories/Widgets	Container	Map Manage-ment	Dynamic Motion	Graphical Represen-tation	Text string	Interactive	Utility	UA Validation
3.3.1	ActiveArea				X		X		B
3.3.2	BasicContainer	X							A
3.3.3	BlinkingContainer	X							
3.3.4	BufferFormat							X	
3.3.5	CheckButton				X	X	X		B
3.3.6	ComboBox				X	X	X		B
3.3.7	Connector							X	A
3.3.8	CursorPosOverlay						X		A
3.3.9	EditBoxMasked				X	X	X		B
3.3.10	EditBoxNumeric				X	X	X		B
3.3.11	EditBoxText				X	X	X		B
3.3.12	GpArcEllipse			X	X				
3.3.13	GpArcCircle			X	X				
3.3.14	GpCrown			X	X				
3.3.15	GpLine			X	X				
3.3.16	GpLinePolar			X	X				
3.3.17	GpRectangle			X	X				
3.3.18	GpTriangle			X	X				
3.3.19	Picture				X				
3.3.20	Label			X	X	X			
3.3.21	LabelComplex				X	X			
3.3.22	MapHorz_ItemList		X		X	X	X		B
3.3.23	MapLegacy		X		X				
3.3.24	MapHorz_Source	X	X				X		A
3.3.25	MapHorz	X	X						A
3.3.26	MaskContainer	X							
3.3.27	Panel	X			X				A
3.3.28	PicturePushButton				X	X	X		B
3.3.29	PictureToggleButton				X	X	X		B
3.3.30	PopUpPanel	X			X		X		
3.3.31	PopUpMenu				X	X	X		
3.3.32	PopUpMenuButton				X	X	X		B
3.3.33	PushButton				X	X	X		B
3.3.34	RadioBox	X							A
3.3.35	RotationContainer	X							
3.3.36	ScrollPanel	X			X		X		B
3.3.37	ScrollList				X	X	X		B
3.3.38	Symbol			X	X				
3.3.39	TabbedPanel	X			X	X			A
3.3.40	TabbedPanelGroup	X			X		X		B
3.3.41	ToggleButton				X	X	X		B
3.3.42	TranslationContainer	X							
	WIDGET EXPANSION SUPPLEMENT 1								
3.4.1	MapGrid		X		X				

	Widget Categories/Widgets	Container	Map Manage-ment	Dynamic Motion	Graphical Represen-tation	Text string	Interactive	**Utility**	**UA Validation**
3.4.2	ExternalSource	X							
3.4.3	MapVert	X	X						**A**
3.4.4	MapVert_Source	X	X				X		**A**
3.4.5	MapVert_ItemList		X		X	X	X		**B**
3.4.6	EditBoxMultiLine				X	X	X		**B**
3.4.7	ComboBoxEdit				X	X	X		**B**
3.4.8	MenuBar	X					X		**A**
	WIDGET EXPANSION SUPPLEMENT 2								
3.5.1	MutuallyExclusive Container	X							**A**
3.5.2	ProxyButton						X		**A**
3.5.3	WatchdogContainer	X							
3.5.4	Slider				X		X		**B**
3.5.5	PictureAnimated				X				
3.5.6	SymbolAnimated			X	X				
3.5.7	SelectionListButton				X	X	X		**B**
	WIDGET EXPANSION SUPPLEMENT 3								
3.6.1	**EditBoxNumeric BCD**				**X**	**X**	**X**		**B**
3.6.2	**CursorRef**							**X**	
3.6.3	**CursorOver**				**X**		**X**		**A**
3.6.4.1	**FocusLink**							**X**	
3.6.4.2	**FocusIn**							**X**	
3.6.4.3	**FocusOut**							**X**	
3.6.5	**SizeToFitContainer**	**X**		**X**	**X**				**A**
3.6.6	**ShuffleToFit Container**	**X**		**X**	**X**				**A**

3.2.3 Container

A Container is a widget that can be referenced as a parent. A Container groups several widgets together. This category of widget is used to design the hierarchical structure of the widget inside the HMI pages.

All objects within a Container become invisible when the Container becomes invisible, as controlled by the Container visible parameter. This should not automatically affect the current value of each widget parameters. The UA is responsible for insuring the coherence of its HMI, for instance the management of EditBoxText inner state. When a container becomes invisible, a contained EditBox cannot stay in its EDIT inner state.

All objects within a container become non-interactive when the Container becomes non-interactive, controlled by the Container enable parameter. This will not automatically affect the current value of each widget parameters.

Widgets placed within Container widgets have their coordinates referenced to the PosX, PosY reference point of the Container. If the Container has no reference point, widgets placed within the Container have their coordinates referenced to the PosX, PosY of the first parent containing a reference point.

Table 3.2.3.1 describes the possible children of Container widgets. Although **an ARINC 661** Layer is not a widget, it has been listed with the Containers because of its capability to be the parent of widgets.

3.2.3.1 Possible Children of Container Widgets

Possible Children of Container Widgets is defined in Table 3.2.3.1.

Table 3.2.3.1 – Possible Children of Container Widgets

Children \ Parents	BasicContainer	BlinkingContainer	Layer	MapHorz	MapHorz_Source	MapVert	MapVert_Source	MaskContainer	MenuBar	MutuallyExclusiveContainer	Panel	PopUpPanel	RadioBox	RotationContainer	ScrollPanel	**SizeToFitContainer**	**ShuffleToFitContainer**	TabbedPanel	TabbedPanelGroup	TranslationContainer	WatchdogContainer
ActiveArea	X		X							X	X	X			X	**X**	**X**	X			X
BasicContainer	X		X							X	X	X			X			X			X
BlinkingContainer	X		X					X		X	X	X		X	X			X		X	X
BufferFormat			X																		
CheckButton	X		X							X	X	X	X		X	**X**	**X**	X			X
ComboBox	X		X							X	X	X			X	**X**	**X**	X			X
Connector				X		X					X							X	X		
CursorPosOverlay	X		X							X	X	X			X			X			X
EditBoxMasked	X		X							X	X	X			X	**X**	**X**	X			X
EditBoxNumeric	X		X							X	X	X			X	**X**	**X**	X			X
EditBoxText	X		X							X	X	X			X	**X**	**X**	X			X
GpArcCircle	X	X	X					X		X	X	X		X	X			X		X	X
GpArcEllipse	X	X	X					X		X	X	X		X	X			X		X	X
GpCrown	X	X	X					X		X	X	X		X	X			X		X	X
GpLine	X	X	X					X		X	X	X		X	X			X		X	X
GpLinePolar	X	X	X					X		X	X	X		X	X			X		X	X
GpRectangle	X	X	X					X		X	X	X		X	X	**X**	**X**	X		X	X
GpTriangle	X	X	X					X		X	X	X		X	X			X		X	X
Label	X	X	X					X		X	X	X		X	X	**X**	**X**	X		X	X
LabelComplex	X		X							X	X	X			X	**X**	**X**	X			X
MapHorz	X		X							X	X	X			X			X			X
MapHorz_ItemList					X																
MapHorz_Source			X	X				X													
MapLegacy					X																
MaskContainer	X		X	X				X		X	X	X			X			X			X
Panel	X		X							X	X	X			X	**X**	**X**	X			X
Picture	X		X							X	X	X			X	**X**	**X**	X			X
PicturePushButton	X		X						X	X	X	X			X	**X**	**X**	X			X
PictureToggleButton	X		X							X	X	X	X		X	**X**	**X**	X			X
PopUpMenu	X		X							X	X	X			X			X			X
PopUpMenuButton	X		X						X	X	X	X			X	**X**	**X**	X			X
PopUpPanel	X		X							X	X				X			X			X

3.0 WIDGET LIBRARY

Children \ Parents	BasicContainer	BlinkingContainer	Layer	MapHorz	MapHorz_Source	MapVert	MapVert_Source	MaskContainer	MenuBar	MutuallyExclusiveContainer	Panel	PopUpPanel	RadioBox	RotationContainer	ScrollPanel	**SizeToFitContainer**	**ShuffleToFitContainer**	TabbedPanel	TabbedPanelGroup	TranslationContainer	WatchdogContainer
PushButton	X		X						X	X	X	X			X	**X**	**X**	X			X
RadioBox	X		X							X	X	X			X			X			X
RotationContainer	X	X	X					X		X	X	X		X	X			X		X	X
ScrollList	X		X							X	X	X			X	**X**	**X**	X			X
ScrollPanel	X		X							X	X	X			X	**X**	**X**	X			X
Symbol	X	X	X					X		X	X	X		X	X			X		X	X
TabbedPanel			X																X		
TabbedPanelGroup	X		X							X	X	X			X			X			X
ToggleButton	X		X							X	X	X	X		X	**X**	**X**	X			X
TranslationContainer	X	X	X					X		X	X	X		X	X			X		X	X
WIDGET EXPANSION SUPPLEMENT 1																					
ComboBoxEdit	X		X							X	X	X			X	**X**	**X**	X			X
EditBoxMultiLine	X		X							X	X	X			X	**X**	**X**	X			X
ExternalSource	X		X					X		X	X			X		**X**	**X**	X		X	X
MapGrid					X		X														
MapVert	X		X							X	X	X			X			X			X
MapVert_ItemList							X														
MapVert_Source			X			X		X													
MenuBar	X		X							X	X	X						X			X
WIDGET EXPANSION SUPPLEMENT 2																					
MutuallyExclusive Container	X		X							X	X	X			X			X			X
ProxyButton	X		X						X	X	X	X			X			X			X
WatchdogContainer	X		X							X	X	X			X			X			X
Slider	X		X							X	X	X			X	**X**	**X**	X			X
PictureAnimated	X		X							X	X	X			X	**X**	**X**	X			X
SymbolAnimated	X		X					X		X	X	X		X	X			X		X	X
SelectionListButton	X		X							X	X	X			X	**X**	**X**	X			X
WIDGET EXPANSION SUPPLEMENT 3																					
EditBoxNumericBCD	**X**		**X**							**X**	**X**	**X**			**X**			**X**			**X**
CursorRef	**X**		**X**		**X**		**X**			**X**	**X**	**X**			**X**	**X**					**X**
CursorOver	**X**		**X**							**X**	**X**	**X**			**X**	**X**	**X**	**X**			**X**
FocusLink	**X**		**X**							**X**	**X**	**X**			**X**			**X**			**X**
FocusIn	**X**		**X**							**X**	**X**	**X**			**X**			**X**			**X**
FocusOut	**X**		**X**							**X**	**X**	**X**			**X**			**X**			**X**
SizeToFitContainer	**X**		**X**							**X**	**X**	**X**			**X**	**X**	**X**	**X**			**X**
ShuffleToFitContainer	**X**		**X**							**X**	**X**	**X**			**X**	**X**	**X**	**X**			**X**

3.2.4 Graphical Representation

Most widgets have a graphical representation. Those that do manifest different appearance aspects according to their StyleSet parameter value. For a given StyleSet, non-interactive widgets have one graphical representation, while interactive widgets may have several graphical representations based on internal state.

3.2.5 Text Strings

Some widgets **and symbols can** contain a string of text (digits, characters, and related symbols). This section describes the available character set for concatenation into a text string.

It also describes the escape sequences principles. The escape capabilities are only available for the following widgets:

- Label**Complex**
- ScrollList

The escape sequences may be embedded in a text string to allow special formatting to occur. The default graphic properties for the text are defined through the "DefaultStyleText" parameter.

For widgets containing a text string, the "MaxStringLength" parameter defines the maximum size of the string; the size is expressed in bytes. This size includes the NULL character that ends the string. For strings containing escape sequences, this size includes all characters, plus the escape sequences, plus the NULL character that ends the string. If several NULL characters end the string (for padding), only the first one is counted inside this length.

Concerning the SetParameter command for modifying a text string, in the A661_ParameterStructure_String or the StringArray_CellStructure, the command structure includes a "StringSize" parameter. This parameter follows the same rule as "MaxStringLength" parameter.

3.2.5.1 Available Character Set

Table 3.2.5.1 defines the characters available for **all** text strings **defined in this specification, (except for the ESC char whose use is described elsewhere)**. The characters in this table are representative of the shapes. It is not intended to define a font. **ARINC 661 characters are stored in a single byte, having codes between 0 and 255 inclusive. Characters labeled "ASCII / UNICODE" correspond both to ASCII and UNICODE. Characters labeled "A661 Extensions" are ARINC 661 specific and correspond to neither ASCII nor UNICODE. ARINC 661 characters not defined in this section are OEM dependent.**

Table 3.2.5.1 – Available Character Set

Control Characters (ASCII / UNICODE)

Hex	Char	Description
h00	NULL	Null character, character for ending a text string.
h0A	LF	Line Feed
h0D	CR	Carriage Return
h1B	ESC	Escape Character, character beginning all escape sequences.

Printing Characters (ASCII/UNICODE)

Hex	Char	Description
h20		space
h21	!	
h22	“	
h23	#	
h24	$	
h25	%	
h26	&	
h27	'	apostrophe
h28	(	
h29	)	
h2A	*	
h2B	+	
h2C	,	comma
h2D	-	dash (minus)
h2E	.	point
h2F	/	slash
h30	0	digit zero
h31	1	
h32	2	
h33	3	
h34	4	
h35	5	
h36	6	
h37	7	
h38	8	
h39	9	
h3A	:	colon
h3B	;	semicolon
h3C	<	
h3D	=	
h3E	>	
h3F	?	
h40	@	
h41	A	
h42	B	
h43	C	
h44	D	
h45	E	
h46	F	
h47	G	
h48	H	
h49	I	

Hex	Char	Description
h4A	J	
h4B	K	
h4C	L	
h4D	M	
h4E	N	
h4F	O	Letter O
h50	P	
h51	Q	
h52	R	
h53	S	
h54	T	
h55	U	
h56	V	
h57	W	
h58	X	
h59	Y	
h5A	Z	
h5B	[	
h5C	\	
h5D	]	
h5E	^	
h5F	_	underscore
h60	`	
h61	a	
h62	b	
h63	c	
h64	d	
h65	e	
h66	f	
h67	g	
h68	h	
h69	i	
h6A	j	
h6B	k	
h6C	l	
h6D	m	
h6E	n	
h6F	o	
h70	p	
h71	q	
h72	r	
h73	s	

Hex	Char	Description
h74	t	
h75	u	
h76	v	
h77	w	
h78	x	
h79	y	
h7A	z	
h7B	{	
h7C	|	
h7D	}	
h7E	~	

Printing Characters (A661 Extensions)

Hex	Char	Description
h80	△	overfly triangle
h81	°	degrees
h82	◇	diamond
h83	□	box
h84	←	Left arrow
h85	→	Right arrow
h86	↑	Up arrow
h87	↓	Down arrow

3.2.5.2 Notation Examples

For example, a text code is designated by ‘G’ or h47

A text string is designated by, as an example: ‘GH12’. This string is the concatenation of the following text codes: ‘G’, ‘H’, ‘1’, ‘2’.

Kxxx : describes a constant value (code or string).

Txxx : describes a type of element. It represents a set of text values (code or string).

“⊗“ is the concatenation symbol.

Examples of concatenation follow:

Concatenation of strings -

If Kyyy = ‘0’ and Kxxx = ‘abc’ then Kyyy⊗Kxxx = ‘0abc’

Concatenation of sets -

If Kyyy = '0' and Txxx = {‘a’, ‘b’, ‘c’} then Kyyy⊗Txxx = {‘0a’, ‘0b’, ‘0c’}

3.2.5.3 Change Style Capabilities

Table 3.2.5.3 lists the available change style capabilities through escape sequence.

Table 3.2.5.3 – Escape Sequence Types

Escape Capabilities	Escape Sequence Type	Description
Foreground Color	TForeColor	Sets the text color. Setting this Escape sequence in the middle of the string will cause all following text to be the new color.
Background Color	TBackColor	Sets the background fill color. Setting this Escape sequence in the middle of the string will cause all following text to have the new background color.
Font	TFont	Sets the font of the text. Setting this Escape sequence in the middle of the string will cause all following text to use the new font.
VideoInv	TVideoInv	Inverse video between the current foreground and background color. The inverse video is applied to characters between this two escape sequences: a Start and an End sequence.
Animation	TAnimation	Animation of text is applied to characters between two escape sequences: Start and End sequence.
Underline	TUnderline	Underline characters capability. Underlining is applied to characters between this two escape sequences: a Start and an End sequence.
Outline	TOutline	An Outline capability. It is the border definition around text.
Bold	TBold	**Bold characters** capability. It is applied to characters between this two escape sequences: a Start and an End sequence.
Crossed	TCrossed	~~Crossed characters~~ capability. It is applied to characters between this two escape sequences: a Start and an End sequence.
Framed	TFramed	Framed characters capability. It is applied to characters between this two escape sequences: a Start and an End sequence.
Repeat Character	TRepeat	Repetition of a set of characters for a specified number of times. It is applied to characters between two escape

Escape Capabilities	Escape Sequence Type	Description
		sequences: Start and End sequence. The repetition number is the first hex value after the start sequence. It is a hex value from 0 to 255 representing the integer number of times to repeat the set of characters.

3.2.5.4 Default Graphic Properties

The “DefaultStyleText” parameter, described in Table 3.3.21-1 (LabelComplex) and Table 3.3.37-1 (ScrollList), indicates if escape sequences are used inside string of the widget. In the case of escape sequence use, it also describes the default background color, foreground color and font for the widget strings.

3.2.5.5 Escape Sequences Description

All escape sequences begin with an “ESC” character, shown in Table 3.2.5.5-1, Escape Sequences Description. An Escape Identifier follows the ESC character (values from h40 to h51 **as defined in Table 3.2.5.5-2**) and any specific parameters required by the sequence (designated by Tvalue). Some escape sequence will apply to the following characters, for instance TForeColor, while some escape sequences will apply between the start and end sequences, for instance TVideoInv.

Escape Sequences Descriptions are defined in Table 3.2.5.5-1.

Table 3.2.5.5-1 – Escape Sequences Description

Type	Starting Sequence	Ending Sequence	Escape Sequence	Size (bits)
TOutline	-	-	ESC⊗Koutline⊗Tvalue0	24
TBackColor	-	-	ESC⊗KbackColor⊗Tvalue1	24
TForeColor	-	-	ESC⊗KforeColor⊗Tvalue1	24
TFont	-	-	ESC⊗Kfont⊗Tvalue2	24
TVideoInv	ESC⊗KvideoInv_B	ESC⊗KvideoInv_E		16
TAnimation	ESC⊗Kanimation_B	ESC⊗Kanimation_E		16
TUnderline	ESC⊗Kunderline_B	ESC⊗Kunderline_E		16
TBold	ESC⊗Kbold_B	ESC⊗Kbold_E		16
TCrossed	ESC⊗Kcrossed_B	ESC⊗Kcrossed_E		16
TFramed	ESC⊗Kframed_B	ESC⊗Kframed_E		16
TRepeat	ESC⊗Krepeat_B⊗P1	ESC⊗Krepeat_E		24 and 16

Where:

P1**:** is a hex value from 0 to 255 representing the integer number of times to repeat the set of characters embedded between the escape sequences: “ESC⊗KRepeat_B⊗P1” and “ESC⊗KRepeat_E”

Tvalue0: Standard for Outline

Bit 1 = T (line on Top)
Bit 2 = B (line on Bottom)
Bit 3 = L (line on Left)
Bit 4 = R (line on Right)

Binary value	Hex code	Description
0011 0000	h30	No Line
0011 0001	h31	T
0011 0010	h32	B
0011 0011	h33	B+T
0011 0100	h34	L
0011 0101	h35	T+L
0011 0110	h36	B+L
0011 0111	h37	T+L+B
0011 1000	h38	R
0011 1001	h39	T+R
0011 1010	h3A	R+B
0011 1011	h3B	B+T+R
0011 1100	h3C	L+R
0011 1101	h3D	L+R+T
0011 1110	h3E	L+B+R
0011 1111	h3F	B+T+L+R (frame)

Tvalue1 = {airframe manufacturer/system integrator-dependent list}

Tvalue2 = {airframe manufacturer/system integrator-dependent list}

Escape Identifers are defined in Table 3.2.5.5-2.

Table 3.2.5.5-2 – Escape Identifier

Escape Identifiers	Value
Koutline	h40
KforeColor	h41
KbackColor	h42
Kfont	h43
KvideoInv_B	h44
KvideoInv_E	h45
Kanimation_B	h46
Kanimation_E	h47
Kunderline_B	h48
Kunderline_E	h49
Kbold_B	h4A
Kbold_E	h4B
Kcrossed_B	h4C
Kcrossed_E	h4D
Kframed_B	h4E
Kframed_E	h4F
Krepeat_B	h50
Krepeat_E	h51

3.2.6 Interactive

Interactive widgets are widgets that have the ability to send an event to their UA. An interactive widget has an Event Structure table attached. Some interactions on these widgets induce an event transmitted to the UA. These widgets will implement different graphical representations according to their state. Refer to Section 3.1.2, Widget States.

3.2.7 Dynamic Motion

UAs have the ability to move dynamic motion widgets at run-time. The parameters PosX, PosY are modifiable at run-time for a dynamic motion widget.

3.2.8 Map Management

The map management category of widgets relates to the management of the symbology inside a map. This section describes a collaboration between widgets fulfilling a map functionality.

There are two types of map: Horizonal and Vertical. In both cases interaction between widgets composing a Map is similar. Following widgets are considered as a part of Map Management group:

For Horizontal Map:

- MapHorz
- MapHorz_Source
- MapHorz_ItemList
- MapGrid

For Vertical Map:

- MapVert
- MapVert_Source
- MapVert_ItemList
- MapGrid

For more detailed information about **a** specific widget refer to the corresponding paragraph in Section 3.3. Widgets as well as a set of predefined symbols are defined at definition time. Number and position of symbols vary at runtime.

Horizontal maps have been used in avionics systems for a considerable amont of time. More recently, vertical maps have been introduced. For this reason more in depth analysis is performed for the horizontal map.

3.2.8.1 Horizontal Map Management

A typical example of horizontal map management widgets is the navigation display format.

A MapHorz_ItemList contains a list of items to be drawn. The type of each item inside the MapHorz_ItemList can be modified at run-time, which makes the list dynamic. A set of parameters is associated with each type of item.

COMMENTARY

MapHorz_ItemList could be used in the creation of flight plan or map background symbols from FM Application, and identification of TCAS intruders from TCAS application.

Addressing of a Item inside a MapHorz_ItemList is described Section 3.3.22, MapHorz_ItemList and illustrated in Appendix E, Map Management Tutorial.

MapGrid widget draws map background as a series of rectangles. More details about MapGrid can be found in Section 3.4.1.

Any Item in MapHorz_ItemList has its position expressed in a local coordinate systems, as opposed to the display unit or screen coordinate system. Thus, to display a MapHorz_ItemList in a format image, a transformation into a window reference system is necessary. This transformation is defined in two steps. First, information about type of local coordinate system is contained in MapHorz_Source. Data in MapHorz_ItemList is meaningless without MapHorz_Source MapDataFormat.

Similar in case of MapGrid, IncrementX and IncrementY are in real-world units defined in MapHorz_Source. As a result, each MapGrid and MapHorz_ItemList has to be a child of MapHorz_Source. Second step is to convert known world coordinate system to screen coordinate system. MapHorz allows to convert from real-world units to screen coordinate system. Rationale behind splitting transformation between MapHorz and MapHorz_Source was to allow objects to merge from multiple world coordinate system into one Map Image. As such, several MapHorz_ItemList and MapGrids can be merged in one Map even if they use different world coordinate systems.

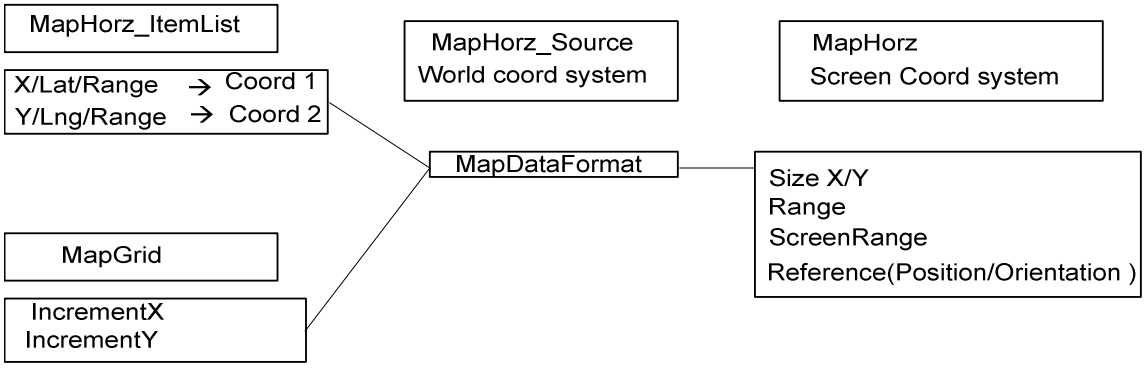


Figure 3.2.8.1 – Coordinate System for MapHorz Widget Management

The UA, which provides MapHorz_ItemList to the CDS, is called a Map User Application. To allow display of the MapHorz_ItemLists in the display area, the Map User Application also provides to the CDS characteristics its MapHorz_ItemLists coordinate system through a widget called MapHorz_Source.

A UA is responsible for passing to the CDS, required reference information for the CDS to perform the merger. In this way, this UA federates all MapHorz_Source data

to enable CDS to perform the merger. This application is called the master application. The master application provides reference information to the CDS through a widget called MapHorz widget.

COMMENTARY

In the ND window case, possible implementation would be ND as the master application while the Map User Applications might be the FMS User Application, TCAS User Application, or other UA.

Different kinds of UAs could be developed to merge data. The description of such UAs is beyond the scope of this document.

3.2.8.1.1 Link Between MapHorz, MapHorz_Source, MapHorz_ItemList and MapGrid

From a hierarchical point of view, illustrated in Figure 3.2.8.1.1, MapHorz widget is a container of MapHorz_Source widgets. It defines reference information for all MapHorz_Sources that it contains. MapHorz_Sources and MapHorz widget are defined by different UAs. Thus, MapHorz_Sources and MapHorz widget are defined in different UALDs or layers. The link between the MapHorz widget and its contained MapHorz_Source will be insured by a Connector widget (refer to Section 3.3.7, Connector). The MapHorz widget can specifically contain only Connector(s) and/or MapHorz_Source(s).

The MapHorz_Source parent can only be the MapHorz widget or the layer. The layer only contains the MapHorz_Source (one or several). One MapHorz_Source can be shared between several MapHorz widgets by using the Connector widget.

The master application manages the visibility of MapHorz_Source from other layers through the connector widget. Indeed the Connector widget has a visibility parameter.

The MapHorz_Source is a container of MapHorz_ItemLists and MapGrid. The MapHorz_Source defines coordinate system characteristics for all its contained MapHorz_ItemLists and MapGrids.

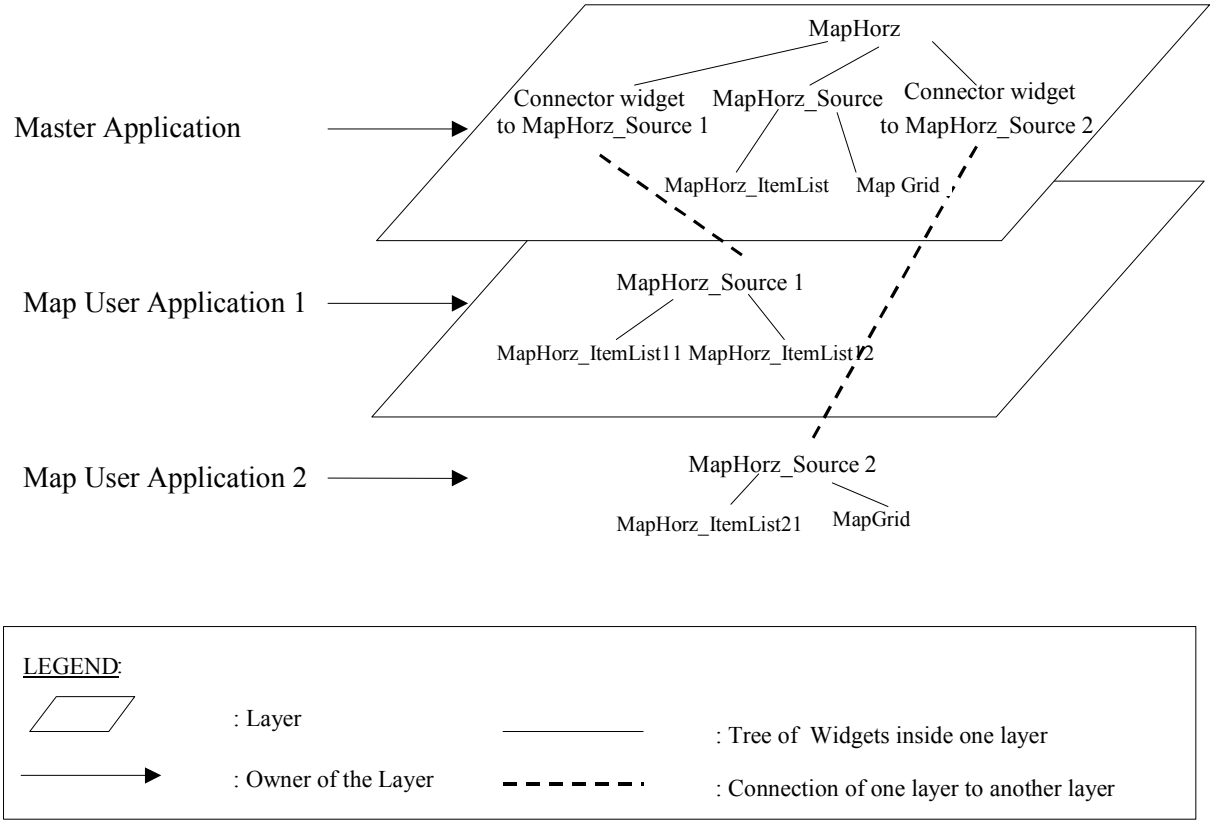


Figure 3.2.8.1.1 – Hierarchical Structure For MapHorz Widget Management

3.2.8.1.2 Parameter Definition for MapHorz and MapHorz_Source

MapHorz and MapHorz_Source widgets hold parameters to assure that, when it is time to draw the map, the CDS will have all the required information.

To define the parameters for MapHorz and MapHorz_Source, the system integrator should review the possible Map User Applications and the kind of display a master application may require. Several examples of master applications are described in the Appendix E, Map Management Tutorial.

The MAPHORZ widget is defined by the following parameters:

MAPHORZ:	
- MAPHORZ X, Y:	Position of the MapHorz widget.
- PRP lat/long:	PRP latitude and longitude.
- PRP X, Y :	PRP position on the display. Value relative to MAPHORZ X,Y.
- True North Angle :	Angle between True North and the Up direction of the display.
- Range Geo referenced;	Range in nm.
- Display equivalent Range :	Range in screen unit.
MAPHORZ_SOURCE:	
- Coordinate System :	Enumerated value. Lat/long is one of these values.

3.2.8.2 Vertical Map Management

A typical example of vertical map management widgets is vertical situation display format. In previous section we described Horizontal map. Vertical map management is similar to horizontal with following substitution for Horz to Vert widgets :

- MapHorz→ MapVert
- MapHorz_Source→ MapVert_Source
- MapHorz_ItemList→ MapVert_ItemList
- MapGrid→ MapGrid

3.2.8.3 Priority Management

The drawing priority between widgets is defined as follows:

- Level 1. The drawing priority between the layer
- Level 2. The drawing priority between the widget inside a layer, as discussed in Section 2.3. The definition order of the widget inside the UALD defines the drawing priority. The last defined widget is the higher priority widget
- Level 3. Then inside a MapHorz_ItemList, the drawing priority is defined by the item order specified by their “ItemIndex” parameter. The higher ItemIndex value has the higher drawing priority

The level 1 and 2 drawing priority are defined statically. The level 3 drawing priority, which is the drawing priority for the item, is defined dynamically at run-time.

The MapHorz_ItemList introduces the notion of container, which is beneficial for managing independently groups of items.

COMMENTARY

The FMS could set in different MapHorz_ItemLists different flight plans, different kinds of background data, etc. The MapHorz_ItemList allows the FMS to group items with different graphical priorities, which correspond to different functional groups.

Correlation between items addressing order and drawing order is developed in Appendix E, Map Management Tutorial.

3.2.8.4 Map Synchronization Number

MapSynchronizationNumber is a parameter of the Map, MapItemList and MapGrid widgets (both horizontal and vertical). The parameter can be set at run-time only. While an initial value cannot be specified in widget creation structures, the initial value for all MapSynchronizationNumber parameters is defined as zero.

As the CDS processes a MapHorz widget, it looks at the map synchronization number. If the MapHorz widget has received a map synchronization number other than zero, then only those MapHorz_ItemList widgets that have the same map synchronization number as the parent MapHorz widget are drawn. The same concept applies to the MapGrid widget, as well as the vertical map widgets.

The value zero is used as a "don't care", both at the MapHorz and the MapHorz_ItemList level. If the MapHorz's map synchronization number is zero (or if a map synchronization number has not been received from the UA), all MapHorz_ItemList widgets are processed, regardless of their map synchronization numbers. Likewise, a MapHorz_ItemList with a map synchronization number of zero will be processed regardless of the number set in the parent MapHorz widget.

The idea is that whenever the map configuration changes (for example range or mode changes), the master ND application can assign and send (through communications independent of ARINC 661) a new map synchronization number to applications controlling item lists that are affected by the change. Once those applications have new map items (matching the new configuration) available, they send a map item list with the assigned map synchronization number. The CDS uses the numbers to determine when item lists are ready to be displayed on a modified map.

A typical (but certainly not the only allowed) sequence of events would look like this:

1. **The Map Master internally decides to update range, scale, rotation, position, etc. of a map and does all necessary related computations and preparations.**

2. **The Map Master increments the Map Synchronization Number in the MapHorz widget for the affected window (this suppresses rendering of any incompatible MapHorz_ItemList) and then sends any updated control parameters to the MapHorz as applicable.**

3. **Then Map Master notifies the UAs of updated control parameters as applicable, through unspecified means (e.g. AFDX broadcast parameters).**

4. **The UA detects the changes in window control parameters and does all necessary related computations and preparations. There is no hurry, because the incompatible MapHorz_ItemLists are no longer being rendered due to the mismatch of the synchronization numbers.**

5. **UA sends an updated map item list to CDS (using 661 run-time messages) and then increments Map Synchronization Number for the updated MapHorz_ItemList widgets to match (thus allowing the widget to be drawn). The UA repeats update/increment sequence for other widgets or widget groups if multiple update cycles are needed for full refresh.**

3.2.9 UA Validation

After the CDS sends a pilot interaction event to the UA, the CDS may suspend further pilot interactions (exact scope is CDS-dependent) to allow the UA time to validate the event. In order for this to occur, all of the following are required:

1. **CDS will need to know which specific widget instances contain events which will need to be validated by the UA. To support this the UA must set the Enable (A661_ENABLE) parameter to**

A661_TRUE_WITH_VALIDATION for each applicable widget instance in either the DF and/or during run-time.

2. **The UA will need to send a notification to let the CDS know when the UA has completed validating the event. This is accomplished when the UA sends the Run-time Modifiable Parameter A661_ENTRY_VALID.**

See Figure 3.2.9 for an example of this UA validation handshake protocol.

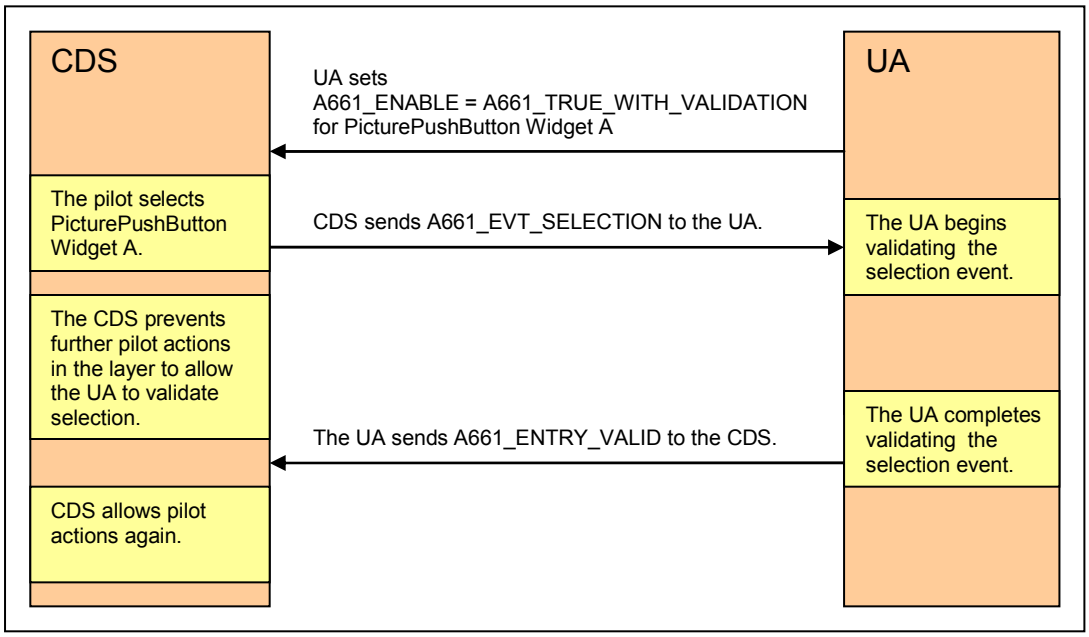


Figure 3.2.9 – Example of UA Validation of a Pilot Selection

Since not all widget types with A661_ENABLE require UA validation, the codes in the UA Validation column in Table 3.2.2-2 indicate the following:

A: This widget type does not support UA validation. Thus, the setting of A661_ENABLE = A661_TRUE_WITH_VALIDATION will be interpreted by the CDS the same as A661_ENABLE = A661_TRUE. This widget type does not support A661_ENTRY_VALID.

B: This widget type supports UA validation and, when the UA has completed validation, the UA will send A661_ENTRY_VALID to the CDS. If during validation the UA determined the pilot action was valid, then A661_ENTRY_VALID will be set to TRUE; if invalid, then set to FALSE.

COMMENTARY

CDS implementations based on versions of this specification prior to Supplement 3 may have been implemented with the assumption that a value of true (A661_TRUE) in the Enable parameter (A661_ENABLE) indicates that the UA has activated the entry validation function. While this is no longer the preferred usage of the Enable parameter, it should be recognized that such implementations exist in the field. The widgets for which this applies are as follows (not an exhaustive list): EditBoxText,

3.3 Widget List

This section describes the characteristics and the interface of ARINC 661widgets. For each widget the definition is divided into parts as follows:

- Definition section
- Widget parameters table
- Creation structure table: CreateParameterBuffer
- Event Structure table
- Run-time modifiable parameter tables

Specific sections are described as follows:

1. This section states the categories of the widget, functional description of the widget and any restrictions to ARINC 661 principles.
2. This section presents the Parameters Table which describes all parameters of the object. These parameters are divided into two categories: “Commonly used parameters” with a reduced description and “Specific parameters” with a complete description. For “commonly used parameter” full descriptions, refer to Section 3.1.3. Also, for “commonly used parameters”, additional information and differences from the norm are underlined.

For each parameter, the following information is presented:

- Name of the parameter
- Possible modifications of parameter by the UA (“change” column)
  - D: parameter set at definition time only through A661_CMD_CREATE command
  - DR : parameter set at definition time through A661_CMD_CREATE and modifiable at run time through A661_CMD_SET_PARAMETER command
  - R : parameter modifiable only at run time through A661_CMD_SET_PARAMETER command
- Description of the parameter

This section describes the format of the exchanges at definition-time, the Create command, as well as at run-time, Widget Event notifications and SetParameter commands for each widget. This description is completed **in S**ection **4**.

The coding format is Big Endian. The types are defined by Table 3.3-1. Fields in the table appear on the bus in the order they are listed.

Table 3.3-1 – Type of Parameters

Type	Standardized Format
uchar	unsigned char coded on 8 bits (used for strings, too)
string	array of uchar Ended by NULL character
long	long integer coded on 32 bits
ushort	unsigned short integer coded on 16 bits
ulong	unsigned long integer coded on 32 bits
float	IEEE 754 format floating point coding on 32 bits (single precision).
fr(x)	Scaled Integer, number of significant bits specified in table. LSB is value in parenthesis (i.e. 'x'), divided by 2-raised-to-the-number-of-bits minus 1. Used for angles. Signed. For example, fr(180) in 16 bits is LSB 0.00549316 fr(180) in 32 bits is LSB 8.381903175442e-8. fr(32768) in 32 bits is LSB 0.0000152587890625.
N/A	Non Applicable

All signed numbers are two's complement form.

In the different structure tables, the structures are built so that:

- 4Bytes and 8Bytes parameters are aligned on 32 bits
- 2Bytes parameters are aligned on 16 bits
- Any UnusedPads are positioned after parameters within a 32 bit word

3. This section presents the "Creation Structure Table" which describes the format of the creation structure: CreateParameterBuffer.

In the Creation Structure Tables, as well as the Event Structure Tables, parameters are grouped together to form words of 32 bits. Each word is separated from other words by a full line. When one word of 32 bits is composed of several parameters, the parts are separated in the table by a dashed line. Refer to the examples in Table 3.3-2.

Table 3.3-2 – Example of Creation Structure

32 bits Words	Name	Type	Size (bits)	Value/Range When Necessary
1	Param1	ushort	16	
	Param2	ushort	16	
2	Param3	ulong	32	
3	Param4	uchar	16	
	Param5	uchar	8	
	Param6	uchar	8	

The parameter order in this table may be different from the order in the Widget parameter table. Indeed, the Widget parameter table describes parameter functional aspect, while the Creation structure table describes the parameter buffer coding aspect.

4. This section presents the “Event Notification Structure” which describes the structure of the events associated with the widget. It describes the events that are able to be sent to the UA by the CDS initiated by a crew member interaction.
5. This section describes the table of parameters modifiable at run time. This table refers to some parameterStructure. This table describes the accessible commands to the UA that manages the widget at run-time.

Some widgets have additional sections to define dedicated data structures.

The following sections define widgets in the Widget Library.

3.3.1 ActiveArea

Categories:
Graphical representation
Interactive

Description:
The ActiveArea is transparent rectangular widget. The ActiveArea may have a graphical representation when this widget is highlighted or when it has the focus. A selection of this widget by a crew member initiates an event notification sent to the owner UA of the widget.

Restriction:
None

ActiveArea Parameters are defined in Table 3.3.1-1.

Table 3.3.1-1 – ActiveArea Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ACTIVE_AREA
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

ActiveArea Creation Structures are defined in Table 3.3.1-2.

Table 3.3.1-2 – ActiveArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ACTIVE_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0

Table 3.3.1-3 defines the specific event sent by the ActiveArea to the owner application.

Table 3.3.1-3 – ActiveArea Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.1-4.

Table 3.3.1-4 – ActiveArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.2 BasicContainer

Categories:
Container

Description:
The BasicContainer has no graphical representation. Its purpose is to group children widgets and to provide a means for managing the visibility and the interactivity of this set of widgets. The contained widgets are positioned with respect to the PosX, PosY of the BasicContainer. It has no clipping capabilities. The position of the BasicContainer can be changed at run-time.

COMMENTARY

BasicContainer is different from a TranslationContainer because it can not be the child of a RotationContainer. BasicContainer can be used to define, at run-time, the position of a button. Translation/Rotation containers are used to translate and rotate graphical primitives or symbols.

Restriction:
N/A

BasicContainer Parameters are defined in Table 3.3.2-1.

Table 3.3.2-1 – BasicContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_BASIC_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point

BasicContainer Creation Structure is defined in Table 3.3.2-2.

Table 3.3.2-2 – BasicContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_BASIC_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	

The BasicContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are:

Basic Container Runtime Modifiable Parameters are defined in Table 3.3.2-3.

Table 3.3.2-3 – BasicContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes

3.3.3 BlinkingContainer

Categories:
Container

Description:
A BlinkingContainer is intended to apply blinking behavior to a group of widgets.

Restriction:
N/A

BlinkingContainer Parameters are defined in Table 3.3.3-1.

Table 3.3.3-1 – BlinkingContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_BLINKING_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Specific parameters		
BlinkingType	DR	Type of blinking (appearance to be defined by the aircraft OEM). Value of zero means no blinking. The definition of all other 255 values is determined by OEM.

BlinkingContrainer Creation Structures is defined in Table 3.3.3-2.

Table 3.3.3-2 – BlinkingContainer Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_BLINKING_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
BlinkingType	uchar	8	
Visible	uchar	8	A661_FALSE A661_TRUE

The BlinkingContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.3-3.

Table 3.3.3-3 – BlinkingContainer Runtime Modifiable Parameters Table

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
BlinkingType	uchar	8	A661_BLINKING_TYPE	A661_ParameterStructure_1Byte

3.3.4 BufferFormat

Categories:
None

Description:
The objective of this widget is to provide a means for grouping data from different widgets (but one layer) in one buffer to reduce overhead. For example, rather than sending <layer id><widget id><parameter id><value><layer id><widget id><parameter id><value><layer id><widget id><parameter id><value><layer id><widget id><parameter id><value>, a supplier can instead define the structure:

<widget id><parameter id>
<widget id><parameter id>
<widget id><parameter id>

and at run-time provide just <layer id><widget id><value><value><value>, which is much more compact. The <widget id> is that of the “BufferFormat” widget.

The buffer structure is fixed at definition time through the BufferStructure parameter. The maximum size of the buffer of values is a function of the number and the nature of the parameters. This buffer structure contains a set of parameter modifiable at run-time. The CDS will perform a set on each parameter identified in the structure.

The widgets referenced in this BufferFormat widget must be defined in the Definition File before the BufferFormat widget. Uses for the BufferFormat include initialization of a layer, and refresh of a large number of widgets at the same time.

Restrictions:

- The BufferFormat can only be the child of a layer
- The BufferFormat can not contain “Definition Only” parameters
- The BufferFormat can not contain parameters which are used inside one of the following structures

  A661_ParameterStructure_Buffer
  A661_ParameterStructure_BufferOfItems
  A661_ParameterStructure_EnableArray
  A661_ParameterStructure_EntryPopUpArray
  A661_ParameterStructure_StringArray

  Indeed, this list corresponds to variable size parameters.
- The BufferFormat can only contain parameters which are used inside one of the following structures

  A661_ParameterStructure_1Byte
  A661_ParameterStructure_2Bytes
  A661_ParameterStructure_4Bytes
  A661_ParameterStructure_String
  A661_ParameterStructure_8Bytes

Variable-size structures cannot be used inside the Buffer parameter of the BufferFormat. The Buffer parameter of the BufferFormat can only be composed of simple parameter values. One exception is the String, which is preceded by two

bytes describing the size of the string in bytes (including the NULL character terminating the string).

BufferFormat Parameters are defined in Table 3.3.4-1.

Table 3.3.4-1 – BufferFormat Parameters Table

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_BUFFER_FORMAT
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget. The only possible parent of the bufferFormat is the layer, therefore ParentIdent Value: 0
Specific parameters		
NumberOf Fields	D	Number of fields in the buffer
BufferStructure	D	Pairs of widgetIdent / ParameterIdent for the value to be sent by the UA through the bufferFormat. The number of pairs is defined by the NumberOfFields. The size of this parameter, in bytes, is (widgetIdent_Size + ParameterIdent_Size)* NumberOfFields
BufferOfParameter	R	Buffer containing the values corresponding to each pair widgetIdent / ParameterIdent.

BufferFormat Creation Structure is defined in Table 3.3.4-2.

Table 3.3.4-2 – BufferFormat Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_BUFFER_FORMAT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
NumberOfFields	ushort	16	
BufferStructure	N/A	32*Number of Fields	Pairs of : WidgetIdent (16 bits) ParameterIdent (16 bits)

The BufferFormat widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.4-3.

Table 3.3.4-3 – BufferFormat Runtime Modifiable Parameters Table

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
BufferOfParameter	N/A	{32}+	A661_BUFFER_OF_PARAM	A661_ParameterStructure_Buffer

3.3.4.1 BufferFormat Alignment

The alignment and padding of data in the BufferFormat is as follows:

- Parameters with size of 4Bytes and 8Bytes are on 32 bits, thus beginning of the value is at the beginning of 32 bits
- Parameters with 2Bytes or String are aligned on 16 bits, thus beginning of the value is at the beginning of 16 bits (beginning of 32 bits or middle of 32 bits).
- No constraint on 1 byte parameters

Figure 3.3.4.1-1 provides examples to illustrate alignment and padding.

The string parameter is defined with the following structure and an alignment on 16 bits:

- 1 * 2Bytes (StringLength) + StringLength * 1Byte + PAD (0 or 8 bits unusedPad to align on 16 bits)
- The stringLength is expressed in Bytes, it describes the number of characters in the string (including the NULL ending the string)

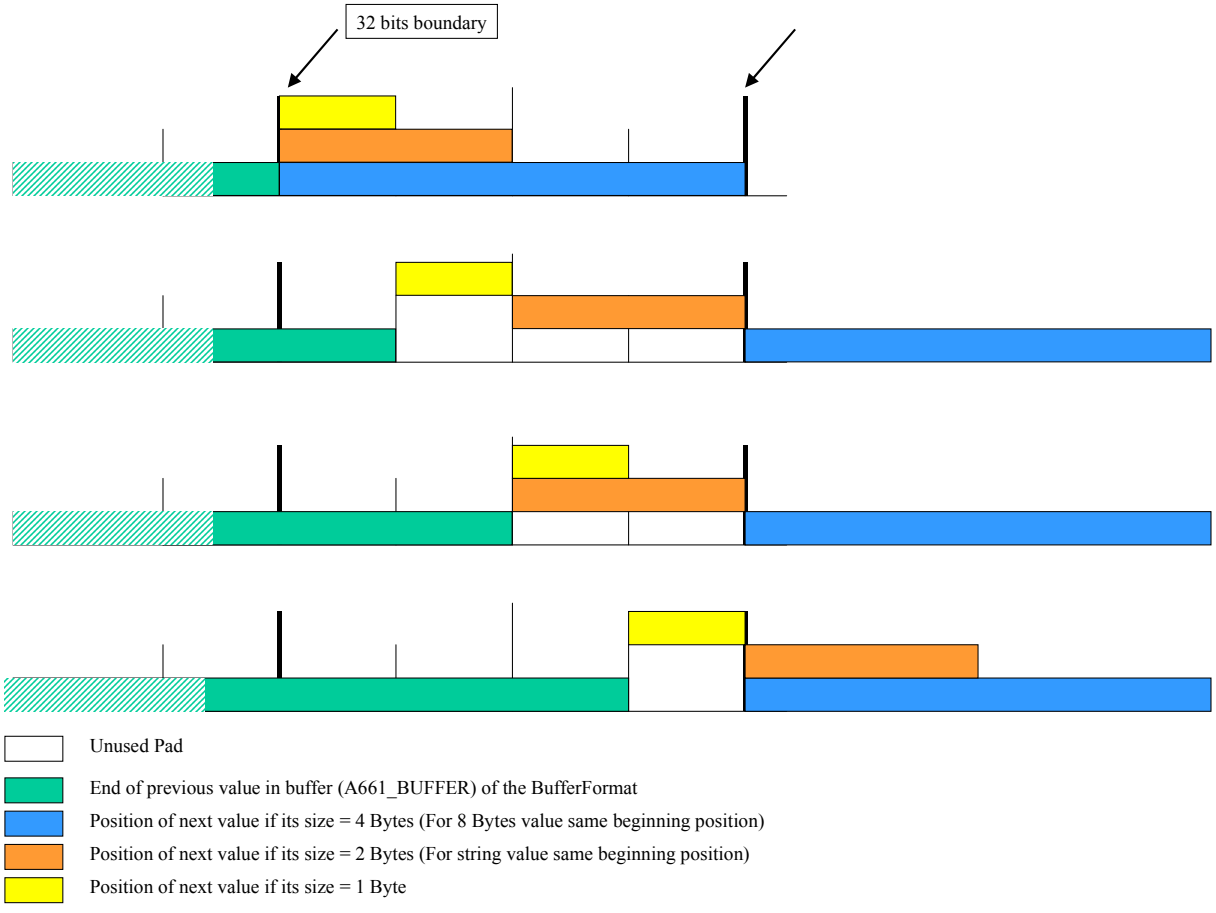


Figure 3.3.4.1-1 – BufferFormat Alignment

3.3.5 CheckButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
A CheckButton allows the crew member to select or not select an option.

Restriction:
N/A

CheckButton Parameters are defined in Table 3.3.5-1.

Table 3.3.5-1 – CheckButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CHECK_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
CheckButtonState	DR	Inner state of the CheckButton: SELECTED UNSELECTED
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
LabelString	DR	Label of the CheckButton
MaxStringLength	D	Maximum length of the label text
Alignment	D	Alignment of the text within the label area of the widget Left Right Center
PicturePosition	D	Position of the CheckBox (picture) with respect to the label within the CheckButton Left Right
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

CheckButton Creation Structure is defined in Table 3.3.5-2.

Table 3.3.5-2 – CheckButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CHECK_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
CheckButtonState	uchar	8	A661_SELECTED A661_UNSELECTED
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
PicturePosition	uchar	8	A661_LEFT A661_RIGHT
UnusedPad	N/A	16	0
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

The specific event sent by the CheckButton to the owner application is defined in Table 3.3.5-3.

Table 3.3.5-3 – CheckButton Event Structures: A661_EVT_STATE_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STATE_CHANGE
CheckButtonState	uchar	8	A661_SELECTED A661_UNSELECTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.5-4.

Table 3.3.5-4 – CheckButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
CheckButtonState	uchar	8	A661_INNER_STATE_CHECK	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.6 ComboBox

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The ComboBox allows a crew member to select one entry within a list. Only the current choice is displayed in the ComboBox area. The number of the current selected entry is held in the SelectedEntry parameter. The complete list of possible Entries is held in a string array (parameter EntryList). The list is displayed upon crew member selection. For example, click on the arrow button associated with the Selected Entry.

Note that SelectingAreaHeight and the SelectingAreaWidth represent the Y and X size of the PopUp part of the ComboBox.

OpeningMode of the ComboBox determines how the ComboBox opens.

The pop-up part of the ComboBox is displayed on top of its containing window and is affected by the clipping area of its containing window.

Restriction:
N/A

ComboBox Parameters are defined in Table 3.3.6-1.

Table 3.3.6-1 – ComboBox Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_COMBO_BOX
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the ComboBox (in the closed mode)
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaHeight	D	Size of the area available to display the entry list
SelectingAreaWidth	D	Size of the area available to display the entry list
OpeningMode	D	Way of combo opening: UP CENTERED DOWN
MaxStringLength	D	Maximum string length for the entries of the list.
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be **less** than **or equal to** MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list.
OpeningEntry	DR	Entry number which is ensured to be visible when the ComboBox is opened. Opening entry is in the range [0; NumberOfEntries] OpeningEntry will be set to 0, if not used.
EntryList [MaxEntryNumber]	DR	String array holding the list of entries.
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

ComboBox Creation Structure is defined in Table 3.3.6-2.

Table 3.3.6-2 – ComboBox Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_COMBO_BOX
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningEntry	ushort	16	
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0
EntryList [NumberOfEntries]	{string}+	{32}+	Each string terminating NULL is used as string separator. The complete string list is followed by zero, one, two or three NULL character(s) to be 32 bits aligned

The specific event sent by the ComboBox to the owner application is defined by Table 3.3.6-3.

Table 3.3.6-3 – ComboBox Event Structures: A661_EVT_SEL_ENTRY_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SEL_ENTRY_CHANGE
EntryNumber	ushort	16	Number of the entry chosen by the crew member

Available SetParameter identifiers and associated data structure are defined in Table 3.3.6-4.

Table 3.3.6-4 – ComboBox Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	16	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
NumberOfEntries	ushort	16	A661_ NUMBER_OF_ENTRIES	A661_ParameterStructure_2Bytes
EntryList [NumberOfEntries]	string[ ]	{32}+	A661_ STRING_ARRAY	A661_ParameterStructure_StringArray
OpeningEntry	ushort	16	A661_OPENING_ENTRY	A661_ParameterStructure_2Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.7 Connector

Categories:
None

Description:
The purpose of this widget is to connect a layer to the container of a different layer. Examples of the use of the Connector widget include TabbedPanelGroup and MapHorz, both of which mix data from several UAs. The action of the Connector widget is functionally like a call to a library routine, or similar reference to a preceding definition. The Connector widget allows another UA to get an image of the referenced widgets. The Connector widget does not imply ownership, copying of the data, or write access. All events associated with the image are handled by the owning application.

Restriction:
The Connector widget capability across physical display surfaces is dependent on system architecture.

Each layer has one priority defined by the current configuration, it does not inherit the priority of its parent layer. In this way, L3 will not inherit the priority of L1 nor L2. Indeed, one UA, for instance, the owner of the L3, can not draw in the graphical layer of another UA in L1 or L2.

The connected layer rendering is affected by the properties of the Container of the connector including: Position, Clipping area, Visible, Enable. Thus, the connected layer has an origin that is defined with respect to the origin of Connector widget parent.

COMMENTARY

If each layer L1 and L2 owns a Connector widget in their UALD reference, the same layer L3, then L1 and L2 should not be interactive at the same time in one given configuration.

Use of the Connector widget may have impact on certification demonstration. Indeed, the Connector widget provides one means for

an UA 1 to manage widgets from an UA 2. For instance, if the UA 1 is level C, then UA 1 it should not manage widgets from UA 2, which is level B.

Connector Parameters are defined in Table 3.3.7-1.

Table 3.3.7-1 – Connector Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CONNECTOR
WidgetIdent	D	Unique identifier of the connector
ParentIdent	D	Identifier of the immediate container of the connector
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
Specific parameters		
Connector Reference	D	Reference of the Connector. It is used to resolve the link with the connected layer. The resolution of the link between the connector and the layer is a configuration issue. All events generated by the widgets of the child layer are still handled by the owning application of this layer.

ConnectorCreation Structure is defined in Table 3.3.7-2.

Table 3.3.7-2 – Connector Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_CONNECTOR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Connector Reference	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
UnusedPad	N/A	16	0

Note: the order of the Visible and Enable parameters for this widget are reversed compared to most other widgets.

The Connector widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.7-3.

Table 3.3.7-3 – Connector Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte

3.3.8 CursorPosOverlay

Categories:
Interactive

Description:
A CursorPosOverlay widget consists of a defined area of the display. The distinguishing characteristic of a CursorPosOverlay is that the reportable event is the current cursor pointer position relative to the CursorPosOverlay. The event is reported on upon selection by a crewmember with a click or keyboard selection.

Restriction:
N/A

CursorPosOverlay Parameters are defined in Table 3.3.8-1.

Table 3.3.8-1 – CursorPosOverlay Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CURSOR_POS_OVERLAY
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Enable	DR	Ability of the widget to generate events.
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget

CursorPosOverlay Creation is defined in Table 3.3.8-2.

Table 3.3.8-2 – CursorPosOverlay Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_CURSOR_POS_OVERLAY
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
UnusedPad	N/A	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	

The specific event sent by the CursorPosOverlay to the owner application is defined in Table 3.3.8-3.

Table3.3.8-3 – CursorPosOverlay Event Structure Tables: A661_EVT_CURSOR_POS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_POS_CHANGE
UnusedPad	N/A	16	0
X	long	32	X position of the cursor with respect to the PosX of the widget
Y	long	32	Y position of the cursor with respect to the PosY of the widget

Available SetParameter identifiers and associated data structure are defined in Table 3.3.8-4.

Table 3.3.8-4 – CursorPosOverlay Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte

3.3.9 EditBoxMasked

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The Masked edit box is an extension of the Text edit box. The difference with the basic Text edit box is that some characters are not modifiable by the crew member. The characters that are not able to be modified are specified by the UA by setting the “alpha mask” parameter and the “numeric mask” parameters to 0.

If a character is only numerical [0..9] , the masks for this character are 1 for numeric mask and 0 for alpha mask. If a character is only alphabetic, i.e., all the printable characters defined in Table 3.2.5.3-1 - Available Character Set excepted [0..9] and SPACE, the masks for this character are 0 for numeric mask and 1 for alpha mask. If a character is alpha-numeric, the masks for this character are 1 for numeric mask and 1 for alpha mask. The size of this string is limited to 32 characters.

COMMENTARY

When the EditBoxMasked is in edit mode, the CDS may report all modifications done on the value of the edited string and the final confirmed string, or only report the confirmed string (after a crew member validation). This option may be set by the UA through the ReportAllChanges parameter. If ReportAllChanges is True and, after having entered text, the crew member aborts the edit, the CDS should send a specific event to the UA with the former validated LabelString as parameter of the event.

EditBoxMasked Parameters are defined in Table 3.3.9-1.

Table 3.3.9-1 – EditBoxMasked Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_MASKED
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
LabelString	DR	Text of the edit box, this string is limited to 32 characters
StartCursorPos	DR	Start position of cursor in the field when entering edit mode
Alignment	D	Justification of the label text within the edit box area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)

Parameters	Change	Description
EntryValidation	R	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**
AlphaMask	DR	Mask for Alpha character
NumericMask	DR	Mask for Numeric character

EditBoxMasked Creation is defined in Table 3.3.9-2.

Table 3.3.9-2 – EditBoxMasked Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_MASKED
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
AlphaMask	ulong	32	
NumericMask	ulong	32	
StartCursorPos	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UnusedPad	N/A	24	0
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE_ABORTED is defined in Table 3.3.9-3.

Table 3.3.9-3 – EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE is defined in Table 3.3.9-4.

Table 3.3.9-4 – EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxMasked Event Structures: A661_EVT_STRING_CONFIRMED is defined in Table 3.3.9-5.

Table 3.3.9-5 – EditBoxMasked Event Structures: A661_EVT_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CONFIRMED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

Table 3.3.9-6 – EditBoxMasked Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.9-7.

Table 3.3.9-7 – EditBoxMasked Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
StartCursorPos	ushort	16	A661_CURSOR_POS	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
AlphaMask	ulong	32	A661_ALPHA_MASK	A661_ParameterStructure_4Bytes
NumericMask	ulong	32	A661_NUMERIC_MASK	A661_ParameterStructure_4Bytes
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte

3.3.10 EditBoxNumeric

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The EditBoxNumeric widget enables editing a numeric value. A crew member can modify the numeric value using input devices. Since a numeric value is used, the CDS is able to increment the value. The widget can receive a number of incremental values or a numeric key value.

EditBoxNumeric Parameters are defined in Table 3.3.10-1.

Table 3.3.10-1 – EditBoxNumeric Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_NUMERIC
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
Value	DR	Value displayed by the edit box in normal mode

<table>
<tr><th>Parameters</th><th>Change</th><th>Description</th></tr>
<tr><td>FormatString</td><td>DR</td><td>String describing the format of the numeric field. The string is composed of any authorized characters. Some of them will be interpreted to format the value.
FormatString is on the form:
“[*](+)[_|*][#|*](.[#|*][_|*])”
[A] means a set of 0 or more of A
where:
(A) means a set of 0 or 1 of A
A | B means A or B
‘_’ indicates an optional display of the digit extracted from the value. If not defined in the value, a blank character will be displayed.
‘#’ indicates a display of the digit extracted from the value or of the character “0”
‘.’ indicates the separator between integer part and decimal part of the value. Must be unique inside StringFormat.
‘+’ forces the sign character to be present in the final string at the defined position. Note that if this sign character is not present in the FormatString, the final string will be one character longer than the FormatString when the value is negative. In that case, the sign will be added at the beginning of the displayed string.
‘*’ Any other characters (any except one of ‘+’, ‘_’, ‘#’, ‘.’) will be interpreted as a mask character.
Note: ‘_’ is not allowed between ‘#’ and ‘.’

Examples:

<table>
<tr><td>FormatString</td><td>Value</td><td>Displayed string</td></tr>
<tr><td>+_ _ _ _ . _ _</td><td>-123.40</td><td>- 123.4</td></tr>
<tr><td>+_ _ _ _ . # #</td><td>123.40</td><td>+ 123.40</td></tr>
<tr><td>_ _ _ _. # #</td><td>-123.45</td><td>-123.45</td></tr>
<tr><td># # # ° # # ‘ # # ‘’</td><td>402318</td><td>040°23’18’’</td></tr>
<tr><td></td><td>123.40</td><td></td></tr>
</table>
</td></tr>
<tr><td>MaxFormatStringLength</td><td>D</td><td>Maximum size of the FormatString</td></tr>
<tr><td>Tics coarse</td><td>DR</td><td>Coarse increment step for modification of the value with main wheel.</td></tr>
<tr><td>Tics fine</td><td>DR</td><td>Fine increment step for modification of the value with secondary wheel.</td></tr>
<tr><td>StartCursorPos</td><td>DR</td><td>Start position of cursor in field when entering edit mode</td></tr>
<tr><td>Alignment</td><td>D</td><td>Justification of the label text within the edit box area:
CENTER
LEFT
RIGHT</td></tr>
<tr><td>ReportAllChanges</td><td>D</td><td>A661_EDB_CHANGE_CONFIRMED
CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED)
A661_EDB_ALL_CHANGE
CDS will report the edit mode opening
A661_EVT_EDITBOX_OPENED
CDS will report each update from the crew member while in edit mode
(A661_EVT_STRING_CHANGE)
CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED)
CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)</td></tr>
</table>

Parameters	Change	Description
		A661_EDB_OPEN_CLOSE CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**
NumericKeyFlag	D	Ability to change the value with the numerical key TRUE FALSE
MaxLegendStringLength	D	Max string size for the legend (MaxLegendStringLength > 0)
LegendAreaSizeX	D	The X dimension (size) of the legend.
LegendString	DR	Legend associated to the numeric value
LegendPosition	DR	Position of the legend in comparison with the numeric value: Left Right Top Bottom
LegendRemoved	D	The flag defining if the legend is to be removed on entry in the editing mode.
MinValue	DR	Minimum value of the **entry**
MaxValue	DR	Maximum value of the **entry**
CyclicFlag	D	Possibility for cyclic modification (or wraparound) of the value TRUE FALSE

EditBoxNumeric Creation Structure is defined in Table 3.3.10-2.

Table 3.3.10-2 – EditBoxNumeric Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_NUMERIC
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
Value	float	32	
Tics coarse	float	32	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
Tics fine	float	32	
MinValue	float	32	
MaxValue	float	32	
LegendAreaSizeX	ulong	32	
StartCursorPosByte	uchar	8	
MaxFormatStringLength	uchar	8	
MaxLegendStringLength	uchar	8	
LegendPosition	uchar	8	A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_OPEN_CLOSE A661_EDB_ALL_CHANGE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
LegendRemoved	uchar	8	A661_FALSE A661_TRUE
NumericKeyFlag	uchar	8	0
CyclicFlag	uchar	8	
UnusedPad	NA	16	
FormatString	string	8 * string length	
LegendString	string	8 * string length + PAD	Followed by zero, one, two or three extra NULL for alignment on 32 bits.

EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE_ABORTED are defined in Table 3.3.10-3.

Table 3.3.10-3 – EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE are defined in Table 3.3.10-4.

Table 3.3.10-4 – EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumeric Event Structures: A661_EVT_STRING_CONFIRMED are defined in Table 3.3.10-5.

Table 3.3.10-5 – EditBoxNumeric Event Structures: A661_EVT_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CONFIRMED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

Table 3.3.10-6 – EditBoxNumeric Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.10-7.

Table 3.3.10-7 – EditBoxNumeric Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte
Value	float	32	A661_VALUE	A661_ParameterStructure_4Bytes
MinValue MaxValue	Float x 2	32 x 2	A661_MINMAX_VALUES	A661_ParameterStructure_8Bytes
Tics coarse	float	32	A661_TICS_COARSE	A661_ParameterStructure_4bytes
Tics fine	float	32	A661_TICS_FINE	A661_ParameterStructure_4bytes
StartCursorPosByte	ushort	8	A661_CURSOR_POS_BYTE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
LegendString	string	{32}	A661_STRING	A661_ParameterStructure_String
FormatString	String	{32}	A661_FORMAT_STRING	A661_ParameterStructure_String
LegendPosition	uchar	8	A661_LEGEND_	A661_ParameterStructure_1Byte

			POSITION	

3.3.11 EditBoxText

Categories:

- Graphical representation
- Interactive
- Text string

Description:
EditBoxText widget enables displaying a string, which can be modified by a crew-member.
The CDS is responsible to perform the following changes of state:
From NORMAL to EDIT
From ERROR to EDIT
The UA is responsible to perform the other transitions.

EditBox Text Parameters are defined in Table 3.3.11-1.

Table 3.3.11-1 – EditBoxText Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_TEXT
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	Maximum length of the text
LabelString	DR	Text of the edit box
StartCursorPos	DR	Start position of cursor in field when entering edit mode.
Alignment	D	Justification of the label text within the edit box area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report each update from the crew member while in edit

Parameters	Change	Description
		mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

EditBoxTextCreation Structure is defined in Table 3.3.11-2.

Table 3.3.11-2 – EditBoxText Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_TEXT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
UnusedPad	N/A	8	0
LabelString	string	8 * string length	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

		+ Pad	

EditBoxText Event Structures: A661_EVT_STRING_CHANGE_ABORTED is defined in Table 3.3.11-3.

Table 3.3.11-3 – EditBoxText Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxText Event Structures: A661_EVT_STRING_CHANGE are defined in Table 3.3.11-4.

Table 3.3.11-4 – EditBoxText Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxText Event Structures: A661_EVT_STRING_CONFIRMED are defined in Table 3.3.11-5.

Table 3.3.11-5 – EditBoxText Event Structures: A661_EVT_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CONFIRMED
StringLength	ushort	16	
String	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits

Table 3.3.11-6 – EditBoxText Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.11-7.

Table 3.3.11-7 – EditBoxText Runtime Modifiable Parameters

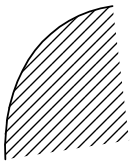
Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
StartCursorPos	ushort	16	A661_CURSOR_POS	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes

3.3.12 GpArcEllipse

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpArcEllipse widget enables the definition of an arc. The arc may be a portion of an ellipse or circle. The arc is defined by a bounding box where a rectangle is specified and the ellipse is drawn touching the rectangle. When the bounding box is a square, the arc will be a circle. The major and minor axes of the ellipse are implicitly along the cardinal directions of the bounding box.



Restriction:
None

GpArcEllipse Parameters are defined in Table 3.3.12-1.

Table 3.3.12-1 – GpArcEllipse Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_ARC_ELLIPSE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
PosX	DR	The X start position of the bounding box (lower left corner).
PosY	DR	The Y start position of the bounding box (lower left corner).
SizeX	DR	The width of the bounding box.
SizeY	DR	The height of the bounding box.
Anonymous	D	Ability to be modified at run-time by the UA.
Specific parameters		
ColorIndex	DR	Color index of the boundary line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present. Halo is a full outline in a contrasting color (typically black) to enhance readability.
Filled	D	If set to True, interior of Arc will be filled.
FillIndex	DR	Fill Pattern index, used if StyleSet allows fill color to be set.
StartAngle	DR	The angle (referenced from the center of the ellipse) that defines the start position of the arc.
EndAngle	DR	The angle (referenced from the center of the ellipse) that defines the end position of the arc.

GpArcEllipse Creation Structure is defined in Table 3.3.12-2.

Table 3.3.12-2 – GpArcEllipse Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_ARC_ELLIPSE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StartAngle	fr(180)	32	
EndAngle	fr(180)	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid Fill Pattern index)
Halo	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

The GpArcEllipse widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.12-3.

Table 3.3.12-3 – GpArcEllipse Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
SizeX	ulong	32	A661_SIZE_X	A661_ParameterStructure_4Bytes
SizeY	ulong	32	A661_SIZE_Y	A661_ParameterStructure_4Bytes
SizeX SizeY	**long x 2**	**32 x 2**	**A661_SIZE_XY**	**A661_ParameterStructure_8Bytes**
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
FillIndex	uchar	8	A661_FILL_INDEX	A661_ParameterStructure_1Byte
StartAngle	fr(180)	32	A661_START_ANGLE	A661_ParameterStructure_4Bytes
EndAngle	fr(180)	32	A661_END_ANGLE	A661_ParameterStructure_4Bytes

3.3.13 GpArcCircle

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpArcCircle widget enables the definition of a circular arc. The circle is defined by a center and radius.

Restriction:
none

GpArcCircle Parameters are defined in Table 3.3.13-1.

Table 3.3.13-1 – GpArcCircle Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_ARC_CIRCLE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS. Refer to section 3.1.3.3.
PosX	DR	The center X position of the circle.
PosY	DR	The center Y position of the circle.
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
ColorIndex	DR	Color index of the boundary line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present
Filled	D	If set to True, interior of Arc will be filled.
FillIndex	DR	Fill Pattern index, used if StyleSet allows fill color to be set.
Radius	DR	The radius of the circle
StartAngle	DR	The angle (referenced from the center of the circle) that defines the start position of the arc.
EndAngle	DR	The angle (referenced from the center of the circle) that defines the end position of the arc.

GpArcCircle Creation Structure is defined in Table 3.3.13-2.

Table 3.3.13-2 – GpArcCircle Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_ARC_CIRCLE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
StartAngle	fr(180)	32	
EndAngle	fr(180)	32	
Radius	ulong	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid fill index)
Halo	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

The GpArcCircle widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.13-3.

Table 3.3.13-3 – GpArcCircle Runtime Modifiable Parameters

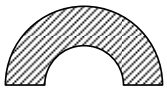
Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
FillIndex	uchar	8	A661_FILL_INDEX	A661_ParameterStructure_1Byte
Radius	ulong	32	A661_RADIUS	A661_ParameterStructure_4Bytes
StartAngle	fr(180)	32	A661_START_ANGLE	A661_ParameterStructure_4Bytes
EndAngle	fr(180)	32	A661_END_ANGLE	A661_ParameterStructure_4Bytes

3.3.14 GpCrown

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpCrown widget enables the definition of a circular filled region. The circle is defined by a center and two radii. The filled area is the area between the radii.



CROWN

Restriction:
None

GpCrown Parameters are defined in Table 3.3.14-1.

Table 3.3.14-1 – GpCrown Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_CROWN
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
PosX	DR	The center X position of the circle.
PosY	DR	The center Y position of the circle.
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
ColorIndex	DR	Color index of the boundary line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present
Filled	D	If set to True, interior of Crown will be filled.
FillIndex	DR	Fill Pattern index, used if StyleSet allows color to be set.
InnerRadius	DR	The radius of the inner circle
OuterRadius	DR	The radius of the outer circle
StartAngle	DR	The angle (referenced from the center of the circle) that defines the start position of the filled arc to be drawn.
EndAngle	DR	The angle (referenced from the center of the circle) that defines the end position of the filled arc to be drawn.

GpCrown Creation Structure is defined in Table 3.3.14-2.

Table 3.3.14-2 – GpCrown Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_CROWN
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
StartAngle	fr(180)	32	
EndAngle	fr(180)	32	
InnerRadius	ulong	32	
OuterRadius	ulong	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid fill index)
Halo	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

The GpCrown widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.14-3.

Table 3.3.14-3 – GpCrown Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
FillIndex	uchar	8	A661_FILL_INDEX	A661_ParameterStructure_1Byte
InnerRadius	ulong	32	A661_INNER_RADIUS	A661_ParameterStructure_4Bytes
OuterRadius	ulong	32	A661_OUTER_RADIUS	A661_ParameterStructure_4Bytes
StartAngle	fr(180)	32	A661_START_ANGLE	A661_ParameterStructure_4Bytes
EndAngle	fr(180)	32	A661_END_ANGLE	A661_ParameterStructure_4Bytes

3.3.15 GpLine

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpLine widget enables the definition of a line. The line is defined in rectangular coordinates by two pairs of X,Y coordinates that define the end points of the line.

Restriction:
None

GpLine Parameters are defined in Table 3.3.15-1.

Table 3.3.15-1 – GpLine Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_LINE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
ColorIndex	DR	Color index of the line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present
PosXStart	DR	The starting X position of the line.
PosYStart	DR	The starting Y position of the line.
PosXEnd	DR	The ending X position of the line.
PosYEnd	DR	The ending Y position of the line.

GpLine Creation Structure is defined in Table 3.3.15-2.

Table 3.3.15-2 – GpLine Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_LINE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosXStart	long	32	
PosYStart	long	32	
PosXEnd	long	32	
PosYEnd	long	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Halo	uchar	8	A661_FALSE A661_TRUE

The GpLine widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.15-3.

Table 3.3.15-3 – GpLine Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosXStart PosYStart	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosXStart	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosYStart	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
PosXEnd PosYEnd	long x 2	32 x 2	A661_POS_XY2	A661_ParameterStructure_8Bytes
PosXEnd	long	32	A661_POS_X2	A661_ParameterStructure_4Bytes
PosYEnd	long	32	A661_POS_Y2	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte

3.3.16 GpLinePolar

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpLinePolar widget enables the definition of a line. The line is defined by polar coordinates with an X,Y coordinate start position, a line length, and a draw angle.

Restriction:
None

GpLinePolar Parameters are defined in Table 3.3.16-1.

Table 3.3.16-1 – GpLinePolar Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_LINE_POLAR
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
ColorIndex	DR	Color index of the line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present
PosXStart	DR	The starting X position of the line.
PosYStart	DR	The starting Y position of the line.
RotationAngle	DR	Angle at which the line is drawn.
LineLength	DR	The length of the line in millimeters.

GpLinePolar Creation Structure is defined in Table 3.3.16-2.

Table 3.3.16-2 – GpLinePolar Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_LINE_POLAR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosXStart	long	32	
PosYStart	long	32	
RotationAngle	fr(180)	32	
LineLength	ulong	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Halo	uchar	8	A661_FALSE A661_TRUE

The GpLinePolar widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.16-3.

Table 3.3.16-3 – GpLinePolar Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosXStart PosYStart	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosXStart	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosYStart	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
RotationAngle	fr(180)	32	A661_ROTATION_ANGLE	A661_ParameterStructure_4Bytes
LineLength	ulong	32	A661_LINE_LENGTH	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte

3.3.17 GpRectangle

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpRectangle widget enables the definition of a rectangle. The primitive defines the start position and the width and height of the rectangle.

Restriction:
None

GpRectangle Parameters are defined in Table 3.3.17-1.

Table 3.3.17-1 – GpRectangle Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_RECTANGLE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
PosX	DR	The X start position of the rectangle (lower left corner).
PosY	DR	The Y start position of the rectangle (lower left corner).
SizeX	DR	The width of the rectangle.
SizeY	DR	The height of the rectangle.
Anonymous	D	Ability to be modified at run-time by the UA.
Specific parameters		
ColorIndex	DR	Color index of the boundary line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present.
Filled	D	If set to True, interior of Rectangle will be filled.
FillIndex	DR	Fill pattern index, used if StyleSet allows fill color to be set.

GpRectangle Creation Structure is defined in Table 3.3.17-2.

Table 3.3.17-2 – GpRectangle Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_RECTANGLE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid Fill Pattern index)
Halo	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

The GpRectangle widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.17-3.

Table 3.3.17-3 – GpRectangle Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
SizeX	ulong	32	A661_SIZE_X	A661_ParameterStructure_4Bytes
SizeY	ulong	32	A661_SIZE_Y	A661_ParameterStructure_4Bytes
SizeX SizeY	**long x 2**	**32 x 2**	**A661_SIZE_XY**	**A661_ParameterStructure_8Bytes**
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
FillIndex	uchar	8	A661_FILL_INDEX	A661_ParameterStructure_1Byte

3.3.18 GpTriangle

Categories:

- Graphical Representation
- Dynamic motion

Description:
The graphical primitive GpTriangle widget enables the definition of a triangle. The primitive defines the three XY coordinate pairs that specify three points of the triangle.

Restriction:
None

GpTriangle Parameters are defined in Table 3.3.18-1.

Table 3.3.18-1 – GpTriangle Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_TRIANGLE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
PosX	DR	The X start position of the triangle (lower left corner).
PosY	DR	The Y start position of the triangle (lower left corner).
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
ColorIndex	DR	Color index of the boundary line, used if StyleSet allows color to be set.
Halo	D	If set to True, Halo is present
Filled	D	If set to True, interior of Triangle will be filled.
FillIndex	DR	Fill Pattern index, used if StyleSet allows fill color to be set.
PosX2	DR	The X position of the second point of the triangle
PosY2	DR	The Y position of the second point of the triangle
PosX3	DR	The X position of the third point of the triangle
PosY3	DR	The Y position of the third point of the triangle

GpTriangle Creation Structure is defined in Table 3.3.18-2.

Table 3.3.18-2 – GpTriangle Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_TRIANGLE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
PosX2	long	32	
PosY2	long	32	
PosX3	long	32	
PosY3	long	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid fill index)
Halo	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

The GpTriangle widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.18-3.

Table 3.3.18-3 – GpTriangle Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
PosX2 PosY2	long x 2	32 x 2	A661_POS_XY2	A661_ParameterStructure_8Bytes
PosX2	long	32	A661_POS_X2	A661_ParameterStructure_4Bytes
PosY2	long	32	A661_POS_Y2	A661_ParameterStructure_4Bytes
PosX3 PosY3	long x 2	32 x 2	A661_POS_XY3	A661_ParameterStructure_8Bytes
PosX3	long	32	A661_POS_X3	A661_ParameterStructure_4Bytes
PosY3	long	32	A661_POS_Y3	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
FillIndex	uchar	8	A661_FILL_INDEX	A661_ParameterStructure_1Byte

3.3.19 Picture

Categories:
Graphical representation

Description:
A Picture widget is a reference to an image available in the CDS. The Picture reference can be modified by the UA. Unlike symbols, a picture can not move or rotate.

Restriction:
N/A

Picture Parameters are defined in Table 3.3.19-1.

Table 3.3.19-1 – Picture Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PICTURE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Anonymous	D	Ability to be modified at run-time by the UA
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
PictureReference	DR	Reference of a picture stored in the CDS

Picture Creation Structure is defined in Table 3.3.19-2.

Table 3.3.19-2 – Picture Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
PictureReference	ushort	16	

The Picture widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.19-3.

Table 3.3.19-3 – Picture Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer Section to 4.5.3)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
PictureReference	ushort	16	A661_PICTURE_REFERENCE	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes

3.3.20 Label

Categories:

- Graphical representation
- Dynamic Motion
- Text string

Description:
A Label widget consists of a non-editable text field at a defined display location. If the label is anonymous, it is not editable (i.e., it can not be modified at runtime by the UA). If it is not anonymous, it can be modified by the UA. However, a label can not be modified by a crew member.

Restriction:
None

Label Parameters are defined in Table 3.3.20-1.

Table 3.3.20-1 – Label Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_LABEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Anonymous	D	Ability to be modified at run-time by the UA
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
LabelString	DR	Text of the label
MaxStringLength	D	Maximum number of character
MotionAllowed	D	Capability to change PosX, PosY, RotationAngle at runtime
RotationAngle	DR	Angle at which symbol is displayed relative to its origin Refer to Angles defined in Section 2.3.4.2)
Font	D**R**	Font of the displayed string**, if StyleSet allows font to be set**
ColorIndex	DR	Applicable color index **for the displayed string, if StyleSet allows color to be set**
Alignment	D	Justification of the label text within the label area BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight

Label Creation Structure is defined in Table 3.3.20-2.

Table 3.3.20-2 – Label Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_LABEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
RotationAngle	fr(180)	32	
StyleSet	ushort	16	
MaxStringLength	ushort	16	
MotionAllowed	uchar	8	A661_FALSE A661_TRUE
Font	uchar	8	
ColorIndex	uchar	8	
Alignment	uchar	8	A661_BOTTOM_CENTER A661_BOTTOM_LEFT A661_BOTTOM_RIGHT A661_CENTER A661_LEFT A661_RIGHT A661_TOP_CENTER A661_TOP_LEFT A661_TOP_RIGHT
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

The Label widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.20-3.

Table 3.3.20-3 – Label Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
RotationAngle	fr(180)	32	A661_ORIENTATION	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
Font	**uchar**	**8**	**A661_FONT**	**A661_ParameterStructure_1Byte**

3.3.21 LabelComplex

Categories:

- Graphical representation
- Text string

Description:
A LabelComplex widget consists of a non-editable text field at a defined display location. If the LabelComplex is anonymous, it is not editable (i.e., it can not be modified at runtime by the UA). If it is not anonymous, it can be modified by the UA. However, a LabelComplex can not be modified by a crew member.

The text string can **contain** embedded escape sequences, refer to Section 3.2.5.5, Escape Sequences Description.

Restriction:
N/A

LabelComplex Parameters are defined in Table 3.3.21-1.

Table 3.3.21-1 – LabelComplex Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_LABEL_COMPLEX
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Anonymous	D	Ability to be modified at run-time by the UA
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
DefaultStyleText	D	NULL character: Escape sequence not used, default value from the CDS used. “TOutLine⊗TBackColor⊗TForeColor⊗TFont”
LabelString	DR	Text of the label
MaxStringLength	D	Maximum number of character
Alignment	D	Justification of the label text within the label area BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight

LabelComplex Creation Structure is defined in Table 3.3.21-2.

Table 3.3.21-2 – LabelComplex Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_LABEL_COMPLEX
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
MaxStringLength	ushort	16	
Alignment	uchar	8	A661_BOTTOM_CENTER A661_BOTTOM_LEFT A661_BOTTOM_RIGHT A661_CENTER A661_LEFT A661_RIGHT A661_TOP_CENTER A661_TOP_LEFT A661_TOP_RIGHT
UnusedPad	N/A	24	0
DefaultStyleText	uchar	96	
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

No event is associated with the LabelComplex widget.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.21-3.

Table 3.3.21-3 – LabelComplex Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes

3.3.22 MapHorz_ItemList

Categories:

- Map management
- Graphical Representation
- Interactive
- Text string

Description:
A MapHorz_ItemList widget represents a group of related graphics. Examples of the use of the MapHorz_ItemList widget is the creation of flight plan, map background symbols, TCAS intruders, etc.

A MapHorz_ItemList must be in a MapHorz_Source container.

A MapHorz_ItemList contains a list of Items to be drawn. This list is of fixed size specified through the maximum number of Items. The type of each Item inside the MapHorz_ItemList can be modified at run-time, which makes the list dynamic. A set of parameters is associated with each type of Item (refer to the “Item Structure” subsection, 3.3.22.2.1, below).

MapHorz_ItemList is different from BufferFormat in that the latter is a list of parameter values for any pre-defined list of widgets, and the former is a list from a limited set of widgets, as well as their parameter values.

One or several items can be modified through a SetParameter command with “BufferOfItems” as Parameter_Ident. An Item should be modified in their entirety, for instance, the latitude of a symbol can not be changed by itself.

Insert and delete operations are not allowed on the list. However, one specific type of Item is NOT_USED. The Item with the NOT_USED type will be ignored, i.e., is they will have no effect on the processing of following items.

Note: This section includes two additional subordinate sections as follows:

Section 3.3.22.1 describes the standardized items and their functionality.

Section 3.3.22.2 describes the A661_ParameterStructure to address the Items.

Restriction:
A MapHorz_ItemList must be in a MapHorz_Source container.
MapHorz_ItemList Parameters are defined in Table 3.3.22-1.

Table 3.3.22-1 – MapHorz_ItemList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_ITEMLIST
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
Specific parameters		
MaxNumberOfItem	D	Maximum number of items that the UA can address under the MapHorz_ItemList.
BufferOfItems	R	Buffer of the Map Items
MapSynchronizationNumber	**R**	**See section 3.2.8.4**
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

MapHorz_ItemList Creation Structure is defined in Table 3.3.22-2.

Table 3.3.22-2 – MapHorz_ItemList Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_ITEMLIST
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfItem	ushort	16	
UnusedPad	N/A	16	0

MapHorz_ItemList Event Structures: A661_EVT_SELECTION are defined in Table 3.3.22-3.

Table 3.3.22-3 – MapHorz_ItemList Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION
Item Index	ushort	16	Index of the item that has been selected. Index from 1 to MaxNumberOfItem.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.22-4.

Table 3.3.22-4 – MapHorz_ItemList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
BufferOfMapItems	N/A	{32}	A661_BUFFER_OF_MAPITEM	A661_ParameterStructure_BufferOfItems Refer to "MapHorz_ItemList A661_ParameterStructure Specifics", Section 3.3.22.2 **and especially Section 3.3.22.2.2** below.
MapSynchronizationNumber	**ushort**	**16**	**A661_MAP_SYNCHRONIZATION_NUMBER**	**A661_ParameterStructure_2Bytes See section 3.2.8.4**
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.22.1 MapHorz_ItemList Standard Items Description

This section describes the MapHorz**_ItemList** item structures.

Table 3.3.22.1 – MapHorz_ItemList Standard Items Description

Name of Item	Function
FILLED_POLY_START	This Item is used to signify the start of a closed, filled polygon definition. It holds X/Y parameters, like LINE_START, and a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items (up to the EndFlag) define the vertices and edges of a polygon that is closed and filled with the indicated fill style.
ITEM_STYLE	For drawing any symbol or line the CDS must apply the last defined ITEM_STYLE in the list. If no ITEM_STYLE has been defined, the CDS will apply a default ITEM_STYLE.
LEGEND	This Item is used to store Legend Strings. Some symbols may contain logic to automatically position legends. LEGEND Items will then follow the SYMBOL Item and carry this legend. Each LEGEND Item can only hold 16 characters including the NULL character. Several LEGEND Item can be used to carry longer strings. CR is recognized as either NextField (For symbols with automatic legend positioning) or as a normal Carriage Return / Line Feed if LEGEND follows a LEGEND_ANCHOR. The last LEGEND Item of a group must have its EndFlag set.
LEGEND_ANCHOR	This Item is used to specify the position of a LEGEND not attached to a symbol.
LEGEND_POP_UP	This Item is a basic LEGEND, but it will appear only when the crew member selects the associated SYMBOL_x Item. Disappearance of the LEGEND_POP_UP is airframe manufacturer/system integrator specification dependent.
LINE_START	This Item is used to signify the start of a line. It holds only X/Y parameters, interpreted by the CDS depending on the MapHorz_Source DataFormat
LINE_SEGMENT	This Item is used to draw a line, using the last defined style in the list, from the previous LINE_xxx End position, to the specified X/Y coordinates. This Item holds EndFlag, set if it is the last item of a line.
LINE_ARC	This Item is used to draw an arc, using the last defined style in the list, from the previous LINE_xxx End position, to the point specified by the three data : (InboundCourse, Radius, CourseChange).

Name of Item	Function
	This Item holds a EndFlag, set if it is the last item of a line.
ITEM_ SYNCHRONIZATION	This item has been defined to attach **frame data** to symbology expressing **the** context of the computation or the rendering for the symbology frame. This data is sent in A661 in order to avoid synchronization issues between the symbology frame and the attached information (e.g. mode, range; MRP …). This item can be used to pass synchronization information from the application owning a MapHorz Item List and the application owning the parent MapHorz.
NOT_USED	This Item is used when the Item is to be discarded by the CDS. There is no effect on subsequent Items interpretation.
SYMBOL_GENERIC	This Item represent the basic symbol which holds X/Y parameters along with a type of symbol and possibly an EndFlag. Some of these types may include an Automatic Legend positioning. In this case, and provided the EndFlag is not set on the symbol, the CDS will interpret the following LEGEND Items as part of the symbol legend. When multiple Fields exist on the symbol, “Carriage Return” will signify to the CDS that a field end is reached.
SYMBOL_ROTATED	Same than SYMBOL_GENERIC except an orientation parameter is added
SYMBOL_CIRCLE	Specific Symbol. It represent a circle of specific radius. Radius is expressed in nm.
SYMBOL_OVAL	Specific symbol. It represents an oval, filled with the indicated fill style, which may be “no fill”.
SYMBOL_RUNWAY	Same as SYMBOL_GENERIC, except orientation and Length parameters are added.
SYMBOL_TARGET	**Same as SYMBOL_GENERIC except part of the symbol can be rotated and part of the symbol has variable length illustrating Velocity / Distance / Altitude.**
TRIANGLE_STRIP _START	**This Item is used to signify the start of a closed, filled polygon defined by a series of triangle strips.**
TRIANGLE_FAN _START	**This Item is used to signify the start of a closed, filled polygon defined by a series of triangle arranged in a fan.**
TRIANGLE_SEGMENT	**Defines a single vertex of a Triangle Strip or Triangle Fan.**
TRIANGLE_SEGMENT _DOUBLE	**Defines two vertices of a Triangle Strip or Triangle Fan.**
TRIANGLE_END	**Defines the last vertex of a Triangle Strip or Triangle Fan.**
TRIANGLE_END _DOUBLE	**Defines the last two vertices of a Triangle Strip or Triangle Fan.**

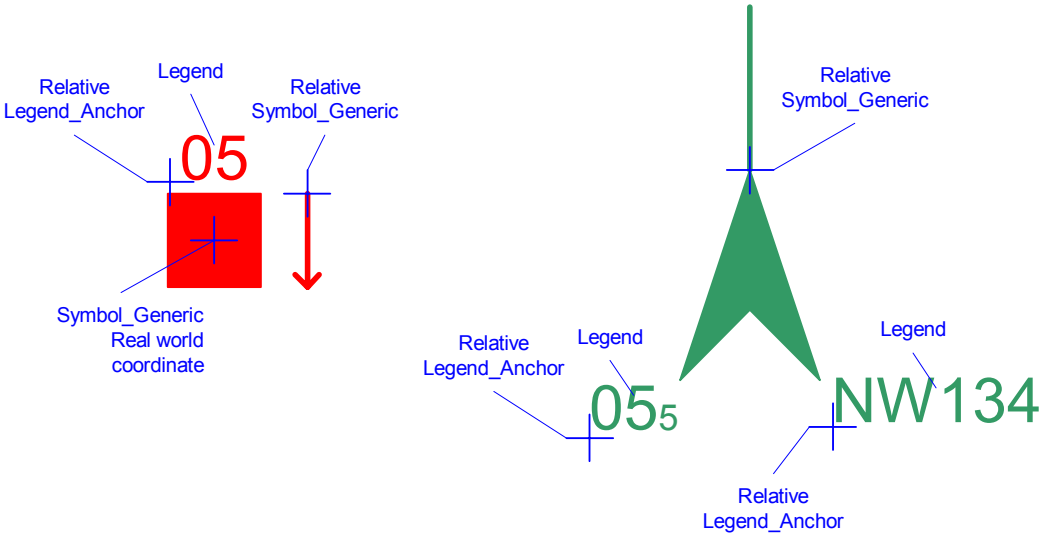
3.3.22.2 MapHorz_ItemList A661_ParameterStructure Specifics

This section describes the A661_ParameterStructure_BufferOfItems **for MapHorz_ItemList**.

Placement of map items is generally determined using a real world coordinate system. In order to support decoration of symbols positioned on the map in real world coordinates relative positioning is also available for several item types. As symbology is placed on a map many times there are other items positioned around it in screen coordinates. The positioning of these relative elements does not change based on the parameters used to calculate coordinate transformations. A relative positioned map item would be used to position symbology using screen coordinates relative to a map item. Symbology that have the RelativePosition parameter associated with them are capable of this relative placement.

In cases where the active areas of two interactive MapHorz_Item widgets overlap, the sending of one or two events will be CDS dependent.

Example of symbol placement using relative screen coordinates:



3.3.22.2.1 Item Structures

All the structures include the same format: three fields for the first 4-byte word. One field is not used on all Items, however it is maintained for consistency.

3.3.22.2.1.1 Item_Style

Item_Style is defined in Table 3.3.22.2.1.1.

Table 3.3.22.2.1.1 – Item_Style

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_STYLE
UnusedPad	N/A	8	0
ItemStyleSet	ushort	16	
UnusedPad	N/A	16	0

3.3.22.2.1.2 Legend_Anchor

Legend_Anchor is defined in Table 3.3.22.2.1.2.

Table 3.3.22.2.1.2 – Legend_Anchor

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapHorz_Source MapDataFormat and RelativePosition)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapSource coordinate system (fixed real LSB depends on MapHorz_Source and RelativePosition)

3.3.22.2.1.3 Legend and Legend_Pop_Up

This Item must follow a XXX_SYMBOL, a LEGEND_ANCHOR or another LEGEND Item. The LegendString can contain special characters, line feed and carriage return. The type of symbol attached to this legend defines the position and the format of this String under control of the CDS. If a LEGEND is followed by other LEGENDs, they should be considered as one unique Legend, possibly including some carriage return and linefeed characters. The full entire LegendString (possibly across multiple Legend MapItems) must have a NULL terminator.

Legend and Legend_Pop_Up is defined in Table 3.3.22.2.1.3.

Table 3.3.22.2.1.3 – Legend and Legend_Pop_Up

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND A661_LEGEND_POP_UP
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendString	{uchar}+ (not 'string')	{32}+	Max 16 characters including NULL and pad Followed by zero, one, two or three extra NULL for alignment of 32 bits. The paragraph above defines the proper string termination.

3.3.22.2.1.4 Line_Start

Line_Start is defined in Table 3.3.22.2.1.4.

Table 3.3.22.2.1.4 – Line_Start

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_START
UnusedPad	N/A	8	0
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)

3.3.22.2.1.5 Line_Segment

Line_Segment is defined in Table 3.3.22.2.1.5.

Table 3.3.22.2.1.5 – Line_Segment

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_SEGMENT
EndFlag	uchar	8	A661_TRUE A661_FALSE
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)

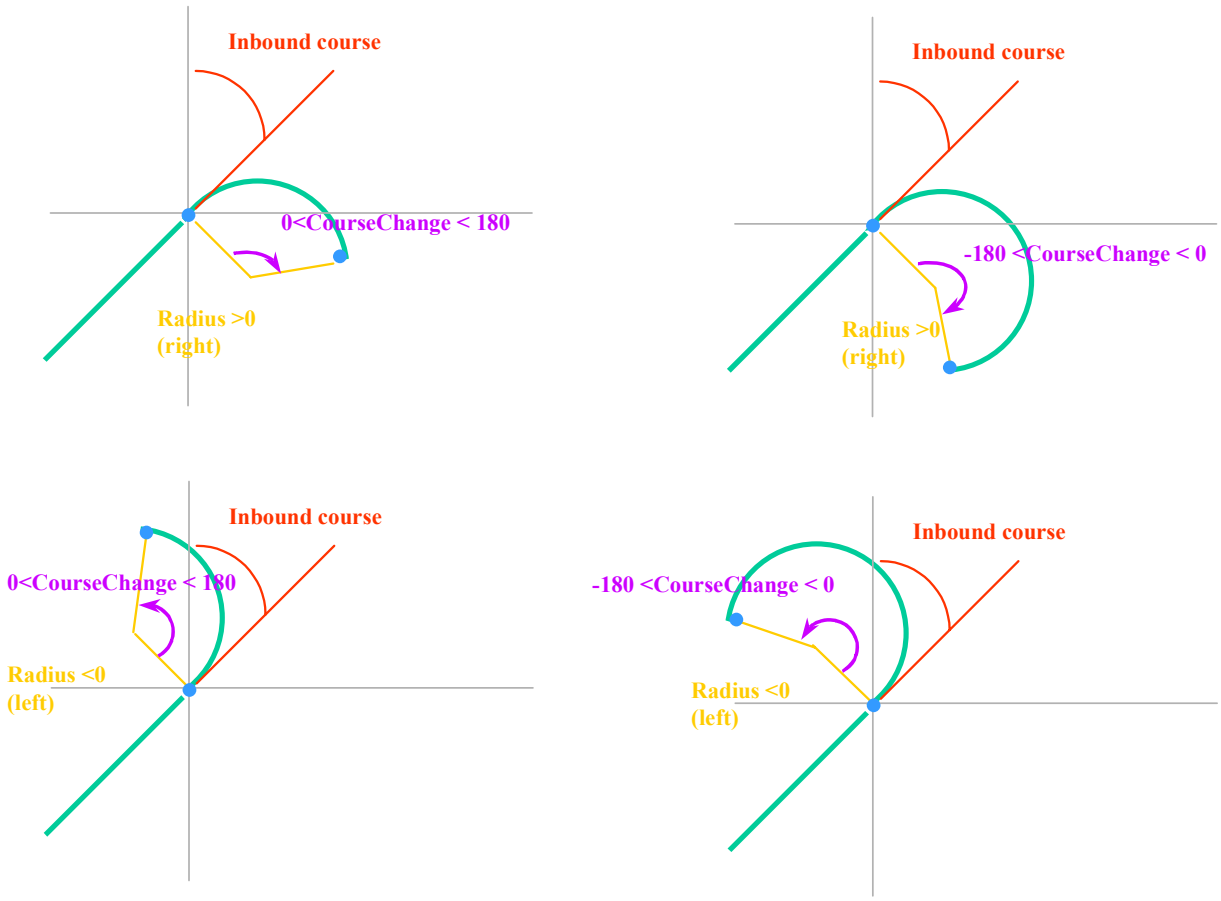
3.3.22.2.1.6 Line_Arc

Line_Arc is defined in Table 3.3.22.2.1.6.

Table 3.3.22.2.1.6 – Line_Arc

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_ARC
EndFlag	uchar	8	A661_TRUE A661_FALSE
InboundCourse	fr(180)	32	A positive InboundCourse indicates a clockwise inbound course with respect to true north (angle$\in$ [0;180]). A negative InboundCourse indicates a counter clockwise inbound course with respect to true north (angle= |InboundCourse| $\in$ [0;180]). See figure that follows this table for more

Name	Type	Size (bits)	Value/Range When Necessary
			information.
Radius	fr(32768)	32	Turn radius Positive for clockwise turn (turn right), Negative for anticlockwise turn (turn left) in nautical miles.
CourseChange	fr(180)	32	A positive CourseChange indicates a turn (Right or left depending of radius sign) of less than 180 degrees : the resulting angle is equal to : CourseChange ∈ [0;180]. A negative CourseChange indicates a turn (Right or left depending of radius sign) of more than 180 degrees : the resulting angle is equal to : (360 + CourseChange) ∈ [0;180].



To specify a complete circle, set the CourseChange to the negative LSB represented by fr(180) or 0xFFFFFFFF.

3.3.22.2.1.7 Not_Used

Not _Used is defined in Table 3.3.22.2.1.7.

Table 3.3.22.2.1.7 – Not_Used

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_NOT_USED
UnusedPad	N/A	8	0

3.3.22.2.1.8 Symbol_Generic

Symbol_Generic is defined in Table 3.3.22.2.1.8.

Table 3.3.22.2.1.8 – Symbol_Generic

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_GENERIC
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X / Lat / Range	Scaled Integer	32	First coordinate of symbol center, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat and RelativePosition)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol center, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat and RelativePosition)

3.3.22.2.1.9 Symbol_Circle

Symbol_Circle is defined in Table 3.3.22.2.1.9.

Table 3.3.22.2.1.9 – Symbol_Circle

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_CIRCLE
EndFlag	uchar	8	A661_TRUE A661_FALSE
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Radius	fr(32768)	32	in nautical miles.

3.3.22.2.1.10 Symbol_Rotated

Symbol_Rotated is defined in Table 3.3.22.2.1.10.

Table 3.3.22.2.1.10 – Symbol Rotated

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_ROTATED
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_HOLD_LEFT SYMBOL_HOLD_RIGHT SYMBOL_PROCEDURE_TURN_LEFT SYMBOL_PROCEDURE_TURN_RIGHT SYMBOL_LONG_RANGE_AIRPORT_WITH_RUNWAY
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat and RelativePosition)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat and RelativePosition)
Orientation	fr(180)	32	Orientation of Symbol relative to True North

3.3.22.2.1.11 Symbol_Runway

Symbol_ Runway is defined in Table 3.3.22.2.1.11.

Table 3.3.22.2.1.11 – Symbol_Runway

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RUNWAY
EndFlag	uchar	8	A661_TRUE A661_FALSE
X / Lat / Range	Scaled Integer	32	First coordinate of runway threshold, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol center, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Length	fr(32768)	32	Length of runway (in feet)
Orientation	fr(180)	32	Orientation of Symbol relative to True North

3.3.22.2.1.12 Filled_Poly_Start

There are restrictions on the polygons to be filled. In particular, the number of line segments is limited to three segments (triangle) or four segments (quadrilateral). The vertices must be specified in counter-clockwise order. The polygon must be convex.

If any error is found in the polygon definition, the CDS should send an A661_ERR_SET_ABORTED exception event. The airframe manufacturer/system integrator free data field may include, for example, the ItemIndex to identify the error.

Filled_Poly_Start is defined in Table 3.2.22.2.1.12.

Table 3.2.22.2.1.12 – Filled_Poly_Start

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_FILLED_POLY_START
FillStyleIndex	uchar	8	See paragraph below
X / Lat / Range	Scaled Integer	32	First coordinate of symbol (LSB and units defined by MapHorz_Source)
Y / Lng / Angle / Alt	Scaled Integer	32	Second coordinate of symbol (LSB and units defined by MapHorz_Source)

3.3.22.2.1.12.1 Fill Style Index Values

A Fill Style Index is an unsigned 8-bit value that is used to select a graphic representation (fill style) from a pre-defined table for use in filling an area on a layer. Because fill styles depend heavily on CDS hardware capabilities, and because they are look-and-feel related, they are not further defined in this specification.

COMMENTARY

The actual fill styles used will depend on both the CDS hardware capability and the supplier/airframe manufacturer/system integrator/customer preference for look-and-feel. A fill style may be a solid color fill, a patterned fill, an alpha blend, or other visual attribute.

3.3.22.2.1.13 Symbol_Oval

Symbol_Oval is defined in Table 3.3.22.2.1.13.

Table 3.3.22.2.1.13 – Symbol_Oval

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_OVAL
FillStyleIndex	uchar	8	airframe manufacturer/system integrator dependent
X / Lat / Range	Scaled Integer	32	First coordinate of symbol center (LSB and units defined by MapHorz_Source)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol center (LSB and units defined by MapHorz_Source)
Radius	fr(32768)	32	Half the Major Axis in nautical miles
Axis Ratio	fr(1)	16	Minor Axis divided by Major Axis
Orientation	fr(180)	16	Orientation of Major Axis relative to True North

3.3.22.2.1.14 Item_Synchronization

Table 3.3.22.2.1.14 – Item_Synchronization

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_SYNCHRONIZATION
DataType	uchar	8	ND_MODE_RANGE MRP latitude / longitude …
SynchronizationData 1st word		32	Values are implementation dependent.
SynchronizationData 2nd word		32	Values are implementation dependent.

This item has been defined to attach to a symbology frame data that expressed the symbology computation context or the rendering context. The functional data is attached to the symbology frame though A661 in order to avoid synchronization issue between the symbology frame and the computation contextual information (e.g., mode, range; MRP, etc).

Thus, the Item_synchronization allows the User Application displaying symbology to transmit functional data to the Master Application through ARINC 661. The functional data will be sent through MapHorzItemList/MapVertItemList by the User Application and will be received by Master Application through MapHorz/MapVert Event.

As an illustration, consider the case of a Master Application having the responsibility to turn on/off the visibility of the connector on a User Application symbology layer, when the Mode/Range context is correct/incorrect.

Upon context change (Range/Mode), the Master Application is responsible for checking the contextual data used by the User Application for computing the symbology frame.

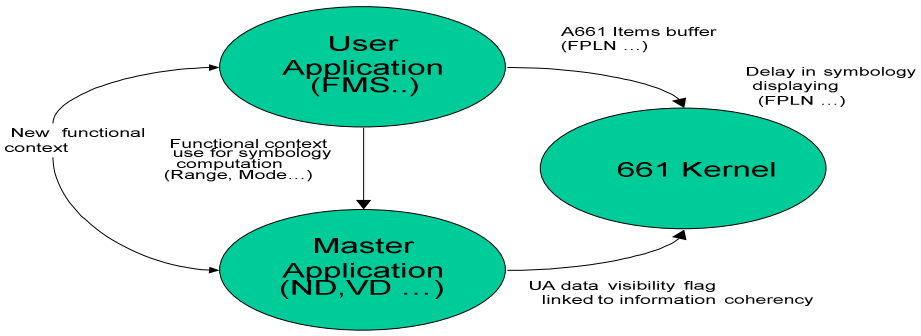


Figure 3.3.22.2.1.14-1 – Map Management Without Item Synchronization

Due to different ways of data transmission, and potential delay on symbology projection before displaying, a de-synchronization may exist between the symbology displaying and the check by the Master Application of the functional context used by the User Application. This de-synchronization has a potential effect on the display.

To avoid cockpit effect due to de-synchronization, Item_Synchronization allows contextual data transmission attached to a symbology frame (like for A702).

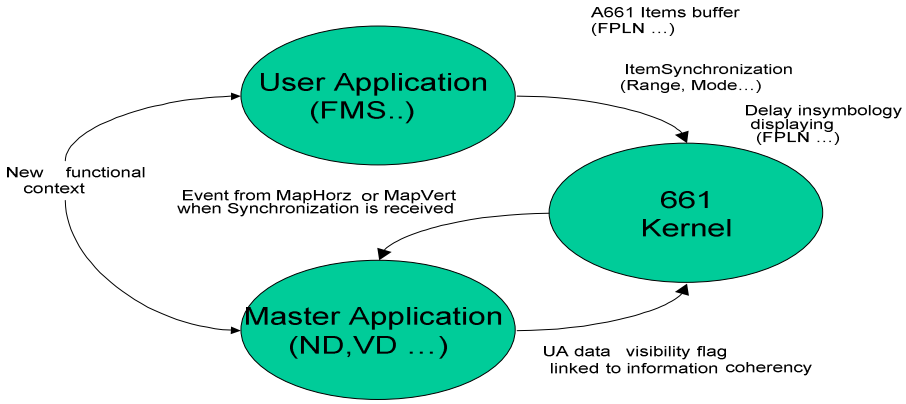


Figure 3.3.22.2.1.14-2 – Map Management With Item Synchronization

Item_synchronization data is sent to the Master Application through a MapHorz/MapVert widget event, when the symbology frame is ready for displaying in order to avoid the de-synchronization effect.

3.3.22.2.1.15 Symbol Target

Table 3.3.22.2.1.15 – Symbol Target

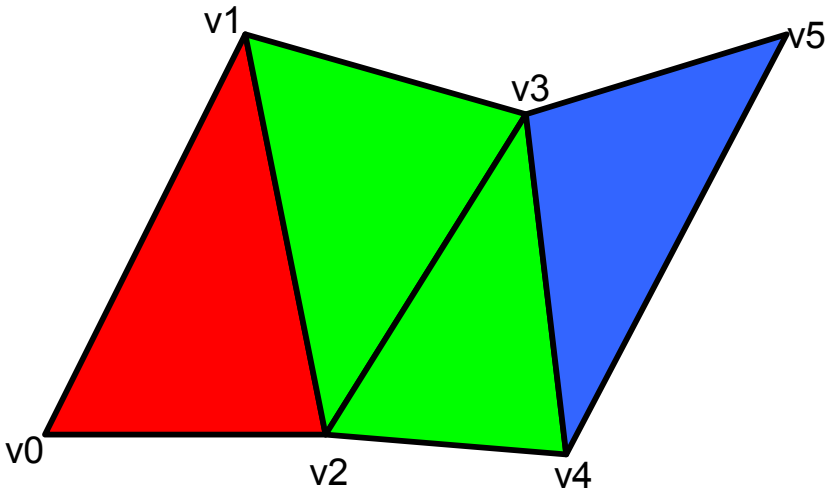
Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_TARGET
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: AIR_FRIEND_TRACK AIR_SUSPECT_TRACK
Length	ushort	16	Velocity / Distance / Altitude illustrated by variable length part of the symbol (Knots / Nautical Miles / Feet)
X / Lat / Range	Scaled Integer	32	First coordinate of symbol center, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol center, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
Orientation	fr(180)	32	Orientation of rotated part relative to True North

3.3.22.2.1.16 Triangle Strip Start

The Triangle Strip Start, Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double MapItemList Items are meant to define triangle strips, similar to those defined in SDL Symbols.

Each Triangle Strip is made up of one Triangle Strip Start, any number of Triangle Segment and Triangle Segment Double items, and one Triangle End or Triangle End Double. The Triangle Segment Double and Triangle End Double items exist to minimize MapItemList size and should be used whenever possible. See the illustration below for more details.

NOTE: Lines shown for illustration only



Red Triangle:
Triangle Strip Start(v0, v1)
Triangle Segment(v2)
Green Triangles:
Triangle Segment Double(v3, v4)
Blue Triangle:
Triangle End(v5)

Figure 3.3.22.2.1.16 – Triangle Strip

Table 3.3.22.2.1.16 – Triangle Strip Start

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_STRIP_START
Pad	NA	8	Unused Pad
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

3.3.22.2.1.17 Triangle Segment

The color of the triangle completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.3.22.2.1.17 – Triangle Segment

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_SEGMENT
FillStyleIndex	uchar	8	Color for the triangle completed by this point
X/Lat/Rng	long	32	First coordinate of vertex
Y/Lng/Brg	long	32	Second coordinate of vertex

3.3.22.2.1.18 Triangle Segment Double

The color of the triangles completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.3.22.2.1.18 – Triangle Segment Double

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_SEGMENT_ DOUBLE
FillStyleIndex	uchar	8	Color for the triangles completed by these points
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

3.3.22.2.1.19 Triangle End

The color of the triangle completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.3.22.2.1.19 – Triangle End

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_END
FillStyleIndex	uchar	8	Color for the triangle completed by this point
X/Lat/Rng	long	32	First coordinate of vertex
Y/Lng/Brg	long	32	Second coordinate of vertex

3.3.22.2.1.20 Triangle End Double

The color of the triangles completed by this item shall be defined by the FillStyleIndex parameter.

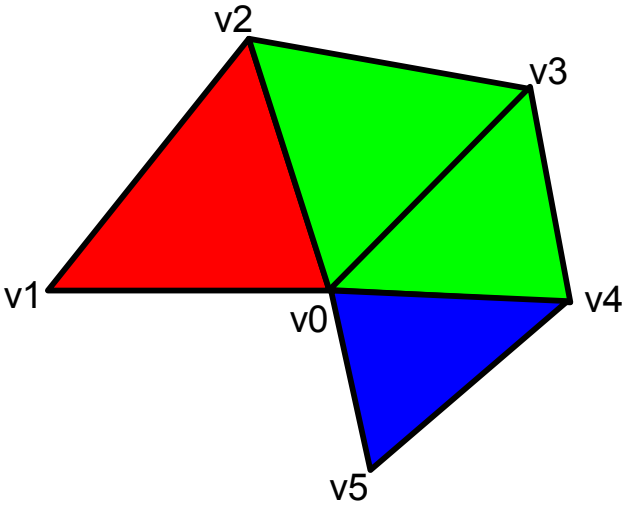
Table 3.3.22.2.1.20 – Triangle End Double

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_END_DOUBLE
FillStyleIndex	uchar	8	Color for the triangles completed by these points
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

3.3.22.2.1.21 Triangle Fan Start

The Triangle Fan Start (along with Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double) MapItemList item is meant to define triangle fans, similar to those defined in SDL Symbols. Each Triangle Fan is made up of one Triangle Fan Start, any number of Triangle Segment and Triangle Segment Double items, and one Triangle End or Triangle End Double. The Triangle Segment Double and Triangle End Double items exist to minimize MapItemList size and should be used whenever possible.

NOTE: Black lines shown for illustration only



Red Triangle:
Triangle Fan Start(v0, v1)
Triangle Segment(v2)
Green Triangles:
Triangle Segment Double(v3, v4)
Blue Triangle:
Triangle End(v5)

Figure 3.3.22.2.1.21 – Triangle Fan

Note: If a triangle is defined as three co-linear points, nothing will be drawn. However, the next triangle will be defined using the first and third points of the previous triangle, as usual.

Table 3.3.22.2.1.21 – Triangle Fan Start

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_FAN_START
Pad	NA	8	Unused Pad
X/Lat/Rng 1	long	32	First Coordinate of first vertex
Y/Lng/Brg 1	long	32	Second Coordinate of first vertex
X/Lat/Rng 2	long	32	First Coordinate of second vertex
Y/Lng/Brg 2	long	32	Second Coordinate of second vertex

Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double are already defined (this was done in conjunction with Triangle Strip Start).

The CDS will process subsequent Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double items appropriately based on the type of triangle start item that preceded them. That is, if a Triangle Fan

Start item begins a sequence of triangle items, the first and third vertex of the previous triangle is used as the base for the triangle that follows. If a Triangle Strip Start item begins a sequence of triangle items, the second and third vertex of the previous triangle is used as the base for the triangle that follows.

3.3.22.2.2 A661_ParameterStructure_BufferOfItems

A661_ParameterStructure_BufferOfItems **as used for MapHorz_ItemList** is defined in Table 3.3.22.2.2.

Table 3.3.22.2.2 – A661_ParameterStructure_BufferOfItems

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_BUFFER_OF_MAPITEM
ClearFlag	1	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
Number of Items	15	Number of Items modified by the command
{ItemStructures}+	{32}+	

3.3.22.3 MapHorz ItemList Interactive Items

This section describes how display applications can specify interactive map items. The CDS highlights interactive items on the map that are close to the cursor. If the user depresses the select button while the interactive item is highlighted, an event is sent by the CDS to the UA. This section covers both MapVert and MapHorz interactive items.

The interactive items available are a subset of those listed in Table 3.3.22.1-1. The constant values for the interactive map items have the same lower seven bits as those listed in Table 3.3.22.1-1 with the eighth bit set. The interactive map items are listed below with their constant values listed in Table 4.6-11.

LINE_ARC_INTERACTIVE
LINE_SEGMENT_INTERACTIVE
LINE_START_INTERACTIVE
SYMBOL_CIRCLE_INTERACTIVE
SYMBOL_GENERIC_INTERACTIVE
SYMBOL_ROTATED_INTERACTIVE
SYMBOL_TARGET_INTERACTIVE
SYMBOL_RUNWAY_INTERACTIVE
FILLED_POLY_START_INTERACTIVE
SYMBOL_OVAL_INTERACTIVE

In cases where the active areas of two interactive MapItem or MapSource widgets overlap, the sending of one or two events will be CDS dependent.

Recommended Behavior:

For line sequences that consist of more than one segment, the application can choose whether highlighting and event reporting is based on individual segments or the entire sequence.

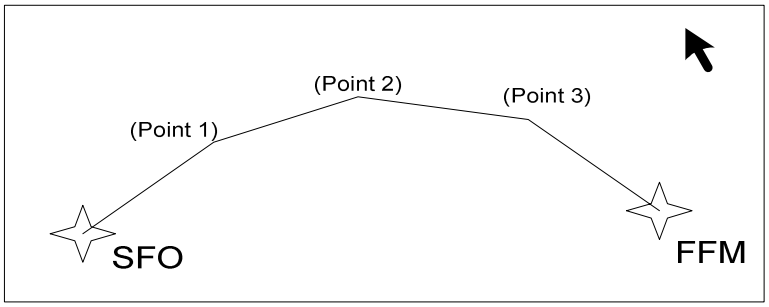


Figure 3.3.22.3-1 – A Line Sequence With Four Straight Line Segments

Figure 3.3.22.3-1 shows a very long line, and for this example it is assumed that the application splits the line into four segments. This could serve the purpose of helping the CDS draw an accurate image of a great-circle line.

If the application intends each line segment to be individually selectable, it can encode the example through the following map item lists (Table 3.3.22.3-1):

Table 3.3.22.3-1 – Map Item List Example for Highlighting and Event Reporting of Individual Segments

Index	Item Type	Data
1	Symbol_Generic	Coordinates of SFO, waypoint symbol
2	Legend	String “SFO”
3	Symbol_Generic	Coordinates of FFM, waypoint symbol
4	Legend	String “FFM”
5	Line_Start	Coordinates of FFM
6	Line_Segment_Interactive	Coordinates of Point 1
7	Line_Segment_Interactive	Coordinates of Point 2
8	Line_Segment_Interactive	Coordinates of Point 3
9	Line_Segment_Interactive	Coordinates of FFM

When the cursor is within highlighting distance of a line segment, only that segment is highlighted (Figure 2). If the pilot clicks on a segment, the A661_EVT_SELECTION event that is sent to the application carries the item index of the segment that was highlighted at the time of the click (item 8 in Table 3.3.22.3-1).

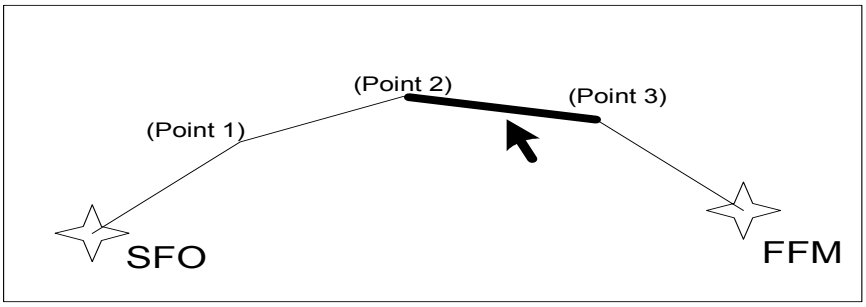


Figure 3.3.22.3-2 – Highlighting and Event Reporting Based on Individual Line Segments

If the application would like to treat the sequence of line segments as a single entity (that just happens to be drawn as separate lines), it may want the entire sequence to be highlighted whenever the cursor is within highlighting distance of any of the line segments, as shown in Figure 3.3.22.3-3 below:

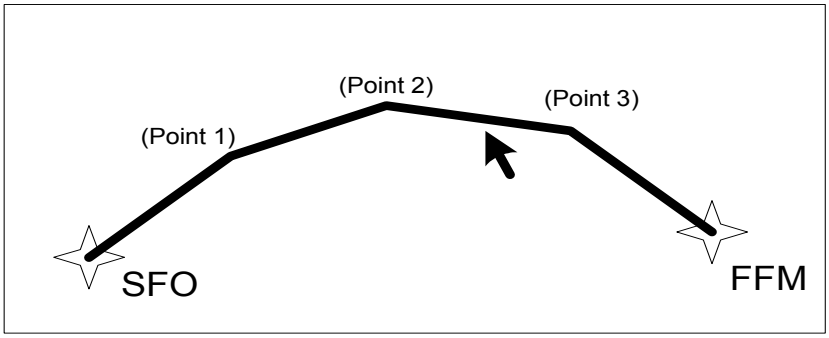


Figure 3.3.22.3-3 – The Entire Segment is Highlighted

As far as event reporting (upon a mouse click) goes, the application can ask the CDS to do one of two things:

1. to send the item index of the individual segment that was highlighted at the time of the click, or
2. to send the item index of the LINE_START_INTERACTIVE item (i.e. the same item regardless of which of its line segments was clicked on).

The application makes this choice by using the LINE_SEGMENT_INTERACTIVE item type if it wants event information for the individual segment, as opposed to the LINE_SEGMENT item type if it wants an event for the overall line sequence. In either case, the line begins with a map item of type LINE_START_INTERACTIVE.

Table 3.3.22.3-2 below shows an example of a line where event reporting is based on the line segment. A click with the cursor as shown in Figure 3.3.22.3-3 would store an item index of 8 in the event structure, as was the case in the previous example (Table 3.3.22.3-1):

Table 3.3.22.3-2 – Map Item List Example for Highlighting Entire Sequence With Event Reporting of Individual Line Segments

Index	Item Type	Data
1	Symbol_Generic	Coordinates of SFO, waypoint symbol
2	Legend	String “SFO”
3	Symbol_Generic	Coordinates of FFM, waypoint symbol
4	Legend	String “FFM”
5	Line_Start_Interactive	Coordinates of FFM
6	Line_Segment_Interactive	Coordinates of Point 1
7	Line_Segment_Interactive	Coordinates of Point 2
8	Line_Segment_Interactive	Coordinates of Point 3
9	Line_Segment_Interactive	Coordinates of FFM

The example in Table 3.3.22.3-3 does not look any different to the pilot. However, when a click on any of the flight plan segment occurs, the item index field in the event message is set to 5 (the index of the LINE_START_INTERACTIVE item) regardless of the particular segment that was clicked on:

Table 3.3.22.3-3 – Map Item List Example for Highlighting and Event Reporting Based on the Entire Line Sequence

Index	Item Type	Data
1	Symbol_Generic	Coordinates of SFO, waypoint symbol
2	Legend	String “SFO”
3	Symbol_Generic	Coordinates of FFM, waypoint symbol
4	Legend	String “FFM”
5	Line_Start_Interactive	Coordinates of FFM
6	Line_Segment	Coordinates of Point 1
7	Line_Segment	Coordinates of Point 2
8	Line_Segment	Coordinates of Point 3
9	Line_Segment	Coordinates of FFM

The following table (Table 3.3.22.3-4) summarizes the CDS behavior. For each line segment, its type (interactive or non-interactive) as well as the type of the beginning of the current line (interactive or non-interactive) determines the highlighting and event behavior.

Table 3.3.22.3-4 – Summary of CDS highlighting and Event Generation

Start of sequence	Current segment	Behavior
Line_Start	Line_Segment	No highlighting, segment is not interactive
Line_Start_Interactive	Line_Segment	Highlight entire sequence; upon click report event for Line_Start_Interactive map item index
Line_Start	Line_Segment_Interactive	Highlighting and event reporting based on individual line segments only
Line_Start_Interactive	Line_Segment_Interactive	Highlight entire sequence; upon click report event for the Line_Segment_Interactive map item that corresponds to the line that the click occurred on

Note: Straight lines were used in the examples. The concept applies to arcs (LINE_ARC and LINE_ARC_INTERACTIVE) the same way.

3.3.23 MapLegacy

Categories:

- Map management
- Graphical Representation

Description:
The MapLegacy widget provides a means for the CDS to be compatible with legacy data formats used with display systems prior to the introduction of ARINC 661. The purpose is to define the means by which the visibility of this kind of data will be managed in relation with other Map UAs. The format of the data and the link to transmit the data depends on the legacy type. Therefore, the data buffer should not be sent through ARINC 661 commands. The CDS and UAs that exchange this type of widget should define together how the CDS processes this data. The MapLegacy is not an interactive widget.

Restriction:
A MapLegacy should be in a MapHorz_Source container.

MapLegacy Parameters are defined in Table 3.3.23-1.

Table 3.3.23-1 – MapLegacy Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAP_LEGACY
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Specific parameters		
ChannelID	D	Identifier of the input stream source reference

MapLegacy Creation Structure is defined in Table 3.3.23-2.

Table 3.3.23-2 – MapLegacy Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_MAP_LEGACY
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
ChannelID	uchar	8	CDS specific
Visible	uchar	8	A661_FALSE A661_TRUE

The MapLegacy widget does not send any event.

The MapLegacy is not modifiable through the A661_CMD_SET_PARAMETER command.

3.3.24 MapHorz_Source

Categories:

- Map management
- Container
- Interactive

Description:
The MapHorz_Source widget is a specialized container. It contains some MapHorz_ItemList widgets to display Items expressed in a common coordinate system.

MapHorz_Source is a widget directly contained by a MapHorz or by **a** Layer, which is directly under the layer in the widget tree. One MapHorz_Source can be shared between several MapHorz widgets using a Connector widget. The format of the data contained by the MapHorz_Source is specified at design time, but the data itself is only available at run time.

MapHorz_Source is an interactive widget. The display area of the MapHorz_Source is the same as the MapHorz. The UA may need to receive the cursor position on a crew member validation with CCD on the MapHorz_Source display area. The MapHorz_Source “EventFlag” parameter provides a means to the Map UA to control the CDS sending this event. The X,Y position sent by the CDS is expressed in MapHorz_Source coordinates.

Restriction:
The MapHorz_Source should be directly under a MapHorz or a Layer widget. When directly attached to a Layer, the layer should not be attached to a window displayed alone.

MapHorz_Source Parameter are defined in Table 3.3.24-1.

Table 3.3.24-1 – MapHorz_Source Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_SOURCE
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
Specific Parameters		
MapDataFormat	D	Format of the data contained by this MapHorz_Source. This parameter defines the coordinate system as well as the kind of transformation to apply on dynamic widgets contained by the MapHorz_Source. See values in the table below.
EventFlag	DR	Type of Event to return: A661_EVENT_NONE: no event will be sent A661_VALIDATION: only SELECTION event will be sent A661_VALIDATION_AND_WHEEL: SELECTION and INCREMENT events will be sent A661_WHEEL: only INCREMENT event will be sent VALIDATION ; it is to indicate if the UA wants to receive the cursor position upon click, expressed in its coordinate system. INCREMENT ; it is to indicate if the user application wants to receive the number of increments upon crew member interaction .

MapHorz_Source Creation Structure is defined in Table 3.3.24-2a.

Table 3.3.24-2a – MapHorz_Source Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
MapDataFormat	uchar	8	A661_MDF_BRG_DIST_ACHDG A661_MDF_DIST_DIST A661_MDF_LAT_LONG A661_MDF_LEGACY
EventFlag	uchar	8	A661_EVENT_NONE A661_VALIDATION A661_VALIDATION_AND_WHEEL A661_WHEEL
UnusedPad	N/A	16	0

MapData Format values are defined in Table 3.3.24-2b.

Table 3.3.24-2b – MapDataFormat Values:

Value	Projection Applied	Alignment of +Y Axis	Origin	Positive Orientation	Units of Measure	LSB
A661_MDF_BRG_DIST_ACHDG	No	aircraft body longitudinal axis	aircraft lat/lng defined in MapHorz	clockwise	X nautical miles Y degrees	X: fr(32768) Y: fr(180)
A661_MDF_DIST_DIST	No	Various	aircraft lat/lng defined in MapHorz	N/A	X and Y: nautical miles	X: fr(32768) Y: fr(32768)
A661_MDF_LAT_LONG	Yes	True North	lat/lng	clockwise	X and Y degrees	X: fr(180) Y: fr(180)
A661_MDF_LEGACY	various	various	various	N/A	various	various

MapHorz_Source Event Structures: A661_EVT_SELECTION_MAP are defined in Table 3.3.24-3.

Table 3.3.24-3 – MapHorz_Source Event Structures*:* A661_EVT_SELECTION_MAP

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION_MAP
UnusedPad	N/A	16	0
X / Lat / Range	Scaled Integer	32	expressed in map source coordinate system
Y / Lng / Bearing	Scaled Integer	32	expressed in map source coordinate system

Table 3.3.24-4a – MapHorz_Source Event Structures*:* A661_EVT_INCREMENT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_INCREMENT
UnusedPad	N/A	16	0
NbOfIncrements	long	32	

Available SetParameter identifiers and associated data structure are defined in Table 3.3.24-4.

Table 3.3.24-4b - MapHorz_Source Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
EventFlag	uchar	8	A661_EVENT_FLAG	A661_ParameterStructure_1Byte

3.3.25 MapHorz

Categories:

- Container
- Map management

Description:
A MapHorz widget consists of a rectangular region on the display, which contains reference information to enable the display of map features in the cockpit. The MapHorz widget enables multiple sources of information with different coordinate systems to be merged into a composite map image.

MapHorz provides information for resolving coordinate system disparities among the map sources. MapHorz also has the responsibility for containing multiple map sources such that the data is merged properly into a composite representation.

Restriction:
None

MapHorz Parameters are defined in Table 3.3.25-1.

Table 3.3.25-1 – MapHorz Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ
WidgetIdent	D	Unique identifier of the widget
Parent Identifier	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
X Pos	D	The X position of the widget reference point (screen coordinate system)
Y Pos	D	The Y position of the widget reference point (screen coordinate system)
SizeX	D	Area size X
SizeY	D	Area size Y
MapSynchronizationNumber	**R**	**See section 3.2.8.4**
Reference coordinate system		
Projection Reference Point Latitude	R	This point is used by the CDS to know what reference should be used to run the projection algorithm. The CDS converts dynamic widget coordinate data into the MapHorz coordinate system. The MapHorz coordinate system is defined by: Reference point: PRP with lat/lng coordinate Reference direction: True North Commentary: For the ND, PRP is the aircraft position for ARC and ROSE mode. For PLAN mode the PRP is a waypoint of the FPLN. In mode PLAN, the PRP can be populated by the FMS through the ND.
Projection Reference Point Longitude	R	
Equivalence between "MapHorz coordinate system" and "MapHorz Screen Coordinate system"		
Screen Reference Point X	DR	X and Y Position of the PRP on the screen. This position is expressed in MapHorz Screen Coordinate System refer to (X Pos, Y Pos)
Screen Reference Point Y	DR	

Parameters	Change	Description
Range	DR	Geo-referenced range
ScreenRange	DR	Distance in screen unit (0.01 mm) equivalent to range
Orientation parameters for latitude/longitude and TCAS coordinate like systems		
Orientation	R	Angle of the True North relative to the up-direction of display at the PRP. (see Reference coordinate system, positive direction: anticlockwise). If PRP Latitude is at a pole, the up-direction of the display should be the meridian identified by PRP Longitude.
AircraftOrientation	R	Orientation of the aircraft relative to the True North (positive direction: clockwise)
AircraftLatitude	R	Latitude of the aircraft
AircraftLongitude	R	Longitude of the aircraft
ProjectionType	D	Indicate which kind of projection will be applied on lat/lng coordinate of dynamic widget. For example: LAMBERT POLAR

MapHorz Creation is defined in Table 3.3.25-2a.

Table 3.3.25-2a – MapHorz Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
Screen Reference Point X	long	32	
Screen Reference Point Y	long	32	
Range	fr(32768)	32	
ScreenRange	ulong	32	
ProjectionType	uchar	8	
UnusedPad	N/A	24	0

MapHorz Event Structures: **the structure of** A661_ EVT_ITEM_ SYNCHRONIZATION **is** defined in Table 3.3.25-2b. This event is initiated by the transmission of an Item_Synchronization in a MapHorz_ItemList. See the definition of the Item_Synchronization in the MapHorz_ItemList for more details.

Table 3.3.25-2b – MapHorz Event Structures: A661_EVT_ITEM_ SYNCHRONIZATION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ITEM_ SYNCHRONIZATION
LinkedIdent	ushort	16	Identifier of the connector identifier link to the layer containing the MapItemList which has received the Item Synchronization. If no connector is used to connect the layer containing the MapItemList with the MapHorz, this field is to be set to identifier of the MapItemList.
DataType	uchar	8	Data type coming from the item synchronization: From MapHorzItemList: ND_MODE_RANGE (for example)
UnusedPad	N/A	24	0
SynchronizationData 1st word		32	Data is implementation dependent.
SynchronizationData 2nd word		32	Data is implementation dependent.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.25-3.

Table 3.3.25-3 – MapHorz Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
Projection Reference Point Latitude	fr(180)	32	A661_PRP_LAT	A661_ParameterStructure_4Bytes
Projection Reference Point Latitude and Longitude	fr(180) x 2	32 x 2	A661_PRP_LAT_LONG	A661_ParameterStructure_8Bytes
Projection Reference Point Longitude	fr(180)	32	A661_PRP_LONG	A661_ParameterStructure_4Bytes
Screen Reference Point X	long	32	A661_PRP_SCREEN_X	A661_ParameterStructure_4Bytes
Screen Reference Point X Screen Reference Point Y	long x 2	32 x 2	A661_PRP_SCREEN_XY	A661_ParameterStructure_8Bytes
Screen Reference Point Y	long	32	A661_PRP_SCREEN_Y	A661_ParameterStructure_4Bytes
Range	fr(32768)	32	A661_RANGE	A661_ParameterStructure_4Bytes
ScreenRange	ulong	32	A661_SCREEN_RANGE	A661_ParameterStructure_4Bytes
Orientation	fr(180)	32	A661_ORIENTATION	A661_ParameterStructure_4Bytes
AircraftLatitude	fr(180)	32	A661_AC_LAT	A661_ParameterStructure_4Bytes
AircraftLongitude	fr(180)	32	A661_AC_LONG	A661_ParameterStructure_4Bytes
AircraftLatitude AircraftLongitude	fr(180) x 2	32 x 2	A661_AC_LAT_LONG	A661_ParameterStructure_8Bytes
AircraftOrientation	fr(180)	32	A661_AC_ORIENTATION	A661_ParameterStructure_4Bytes
MapSynchronizationNumber	**ushort**	**16**	**A661_MAP_SYNCHRONIZATION_NUMBER**	**A661_ParameterStructure_2Bytes See section 3.2.8.4**

3.3.26 MaskContainer

Categories:
Container

Description:
A MaskContainer widget applies a mask to a group of widgets to implement non-rectangular clipping. A mask should be referenced and placed by the Container. Widgets placed within this Container will be affected by the referenced mask.

Restriction:
None

MaskContainer Parameters are defined in Table 3.3.26-1.

Table 3.3.26-1 – MaskContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MASK_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
PosX	D	X position of the mask. Note that this does not reposition widgets contained within the MaskContainer, only the mask itself.
PosY	D	Y position of the mask. Note that this does not reposition widgets contained within the MaskContainer, only the mask itself.
Specific parameters		
MaskReference	DR	Index to a Mask stored in the CDS. See definition of Mask in the Glossary.
MaskEnabled	DR	If set to True, the mask is active and all the widgets contained within the MaskContainer will be affected by the referenced mask. If set to False, the mask is not active and the widgets contained within the MaskContainer will not be affected by the referenced mask.

MaskContainer Creation Structure is defined in Table 3.3.26-2.

Table 3.3.26-2 – MaskContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MASK_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
MaskEnabled	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
MaskReference	ushort	16	
UnusedPad	N/A	16	0

The MaskContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.26-3.

Table 3.3.26-3 – MaskContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
MaskReference	ushort	16	A661_MASK_REFERENCE	A661_ParameterStructure_2Bytes
MaskEnabled	uchar	8	A661_MASK_ENABLED	A661_ParameterStructure_1Byte

3.3.27 Panel

Categories:

- Container
- Graphical representation

Description:
A Panel widget groups several widgets together in a rectangular area with clipping capabilities. Widgets placed within a Panel widget have their coordinates referenced to the PosX, PosY reference point of the Panel.

Restriction:
None

Panel Parameters are defined in Table 3.3.27-1.

Table 3.3.27-1 – Panel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PANEL
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
MotionAllowed	**D**	**Capability to change PosX, PosY, SizeX, SizeY at run time**
PosX	D**R**	The X position of the widget reference point
PosY	D**R**	The Y position of the widget reference point
SizeX	D**R**	The X dimension size (width) of the widget
SizeY	D**R**	The Y dimension size (height) of the widget

Panel Creation Structure is defined in Table 3.3.27-2.

Table 3.3.27-2 – Panel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
MotionAllowed	**uchar**	**8**	**A661_FALSE** **A661_TRUE**
UnusedPad	N/A	**8**	0

No event is associated with the Panel widget.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.27-3.

Table 3.3.27-3 – Panel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX **PosY**	**long x 2**	**32 x 2**	**A661_POS_XY**	**A661_ParameterStructure_8Bytes**
PosX	**long**	**32**	**A661_POS_X**	**A661_ParameterStructure_4Bytes**
PosY	**long**	**32**	**A661_POS_Y**	**A661_ParameterStructure_4Bytes**
SizeX	**ulong**	**32**	**A661_SIZE_X**	**A661_ParameterStructure_4Bytes**
SizeY	**ulong**	**32**	**A661_SIZE_Y**	**A661_ParameterStructure_4Bytes**
SizeXY	**long x 2**	**32 x 2**	**A661_SIZE_XY**	**A661_ParameterStructure_8Bytes**

3.3.28 PicturePushButton

Categories:

- Interactive
- Graphical representation
- Text string

Description:
A PicturePushButton widget is a PushButton including a picture and possibly a string.

Restriction:
None

PicturePushButton Parameters are defined in Table 3.3.28-1.

Table 3.3.28-1 – PicturePushButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PICTURE_PUSH_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	Maximum length of the label text
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelString	DR	Label of the PicturePushButton
Picture Reference	DR	Reference of the picture
PicturePosition	D	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

Picture PushButton Creation Structure is defined in Table 3.3.28-2.

Table 3.3.28-2 – Picture PushButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE_PUSH_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
PictureReference	ushort	16	
MaxStringLength	ushort	16	
PicturePosition	uchar	8	A661_LEFT A661_CENTER A661_RIGHT A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UnusedPad	N/A	8	0
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

Picture PushButton Event Structures: A661_EVT_SELECTION is defined in Table 3.3.28-3.

Table 3.3.28-3 – Picture PushButton Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.28-4.

Table 3.3.28-4 – Picture PushButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
PictureReference	ushort	16	A661_PICTURE_REFERENCE	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.29 PictureToggleButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
A PictureToggleButton widget is a button with two stable states with a picture and possibly text.

Restriction:
None

PictureToggleButton Parameters is defined in Table 3.3.29-1.

Table 3.3.29-1 – PictureToggleButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PICTURE_TOGGLE_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
ToggleState	DR	Inner state of the ToggleButton SELECTED UNSELECTED
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		

Parameters	Change	Description
MaxStringLength	D	Maximum length of the label text
AlternateFlag	D	True: Use of the two string (and two picture) according to the inner state. CDS will change the string if the inner state change False: "AlternateString" (and AlternatePicture) is not used. Only parameter "string" (and Picture) is used for the two inner state
LabelString	DR	Label of the ToggleButton Label used for UNSELECTED state
AlternateLabelString	DR	Label of the ToggleButton Label used for SELECTED state
PictureReference	DR	Picture on the ToggleButton Picture used for UNSELECTED state
AlternatePictureReference	DR	Picture on the ToggleButton Picture used for SELECTED state
Alignment	D	Alignment of the text within the label area of the widget: LEFT RIGHT CENTER
PicturePosition	D	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

PictureToggleButton Creation Structure is defined in Table 3.3.29-2.

Table 3.3.29-2 – PictureToggleButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE_TOGGLE_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
AlternateFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
PictureReference	ushort	16	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
AlternatePictureReference	ushort	16	
PicturePosition	uchar	8	A661_LEFT A661_CENTER A661_RIGHT A661_TOP A661_BOTTOM
ToggleState	uchar	8	A661_UNSELECTED A661_SELECTED
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UnusedPad	N/A	8	0
LabelString	string	8 * string1 length	The string terminating NULL is used as string separator.
AlternateLabelString	string	8 * string2 length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

PictureToggleButton Event Structures: A661_EVT_STATE_CHANGE are defined in Table 3.3.29-3.

Table 3.3.29-3 – PictureToggleButton Event Structures: A661_EVT_STATE_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	**ushort**	16	A661_EVT_STATE_CHANGE
ToggleState	**uchar**	8	A661_UNSELECTED A661_SELECTED
UnusedPad	**N/A**	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.29-4.

Table 3.3.29-4 – PictureToggleButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
ToggleState	uchar	8	A661_INNER_STATE_TOGGLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PictureReference	ushort	16	A661_PICTURE_REFERENCE	A661_ParameterStructure_2Bytes
AlternatePicture Reference	ushort	16	A661_ALTERN_PICTURE_ REFERENCE	A661_ParameterStructure_2Bytes
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
AlternateLabelString	string	{32}+	A661_STRING_ALTERNATE	A661_ParameterStructure_String
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.30 PopUpPanel

Categories:

- Container
- Graphical representation
- Interactive

Description:
PopUpPanel widget should be displayed on top of other layers, but it is affected by clipping area of its parents.

PopUpPanel widget visibility should be managed by the CDS through logic defined by the airframe manufacturer/system integrator.

PopUpPanel widget should not be used as a regular Container. The UA or CDS can define the position of the Container according to the PositionFlag value.

PopUpPanel widget has clipping capability.

Restriction:
PopUpPanel Parameters are defined in Table 3.3.30-1.

Table 3.3.30-1 – PopUpPanel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_POP_UP_PANEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	R	Visibility of the widget**:** **A661_FALSE** **A661_TRUE** **Note:** Widget is not visible at creation time.
StyleSet	DR	Reference to graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
UAPositionFlag	D	TRUE: UA defined position FALSE: Position defined by CDS using MouseClick location
AutomaticClosure	D	TRUE: with the automatic closure upon a click outside the PopUpPanel FALSE: without the automatic closure upon a click outside the PopUpPanel

PopUpPanel Creation Structure is defined in Table 3.3.30-2.

Table 3.3.30-2 – PopUpPanel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_POP_UP_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UAPositionFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticClosure	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	Set to 0 when UAPositionFlag is FALSE.
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0

The specific event sent by the PopUpPanel to the owner application is defined in Table 3.3.30-3.

Table 3.3.30-3 – PopUpPanel Event Structures: A661_EVT_POPUP_PANEL_CLOSED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_POPUP_PANEL_CLOSED
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.30-4.

Table 3.3.30-4 – PopUpPanel Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Byte

3.3.31 PopUpMenu

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The PopUpMenu widget should be displayed on top of other layers, but it is affected by the clipping area of its parents. PopUpMenu is not a Container.

PopUpMenu visibility should be managed by the CDS using logic defined by the airframe manufacturer/system integrator. The UA or CDS can define the position of the Container according to the OpeningMode value.

Restriction:
None

PopUpMenu Parameters are defined in Table 3.3.31-1.

Table 3.3.31-1 – PopUpMenu Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_POP_UP_MENU
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	R	Visibility of the widget: **A661_FALSE** **A661_TRUE** **Note:** Widget is not visible at creation time.
StyleSet	DR	Referenced to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
OpeningMode	D	OPEN_UP: the position (X,Y) is given according to bottom/left point OPEN_DOWN: the position (X,Y) is given according to top/Left point CDS_DEPENDENT: Position defined by CDS using CCD Click location
NumberOfEntries	D	Number of entries in the PopUpMenu. It also includes the graphical separators.
MaxStringLength	D	Maximum length of the text of any one entry including the first NULL character ending the string (in bytes).
StringArray	DR	String attached to one entry Strings composed by only one NULL character will be interpreted as “graphical separator”. The NULL string at the end of the array will be not interpreted.
PictureArray	DR	Picture attached to each entry (if value is NULL, no picture is attached to the corresponding entry).
PopUpIdentArray	D	WidgetIdent for the PopUpMenu attached to one string. WidgetIdent can only refer to another PopUpMenu. If WidgetIdent is NULL, no PopUpMenu is attached to this Entry.
EnableArray	DR	Ability for each Entry on the PopUpMenu: ENABLE DISABLE

PopUpMenu Creation Structure is defined in Table 3.3.31-2.

Table 3.3.31-2 – PopUpMenu Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_POP_UP_MENU
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_DOWN A661_CDS_DEPENDENT
NumberOfEntries	uchar	8	
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
MaxStringLength	ushort	16	Maximum length of the text of one entry
StyleSet	ushort	16	
PopUpIdentArray[NumberOfEntries]	{ushort}+	16 * NumberOf Entries	
PictureArray [NumberOfEntries]	{ushort}+	16 * NumberOfEntries	
EnableArray[NumberOfEntries]	{uchar}+	8 * NumberOf Entries	
StringArray[NumberOfEntries]	{string}+	8 * string lengths + Pad	There are “NumberOfEntries” strings. Each string is ended by character NULL (part of the string used as string separator). The complete string list is followed by zero, one, two or three NULL character(s) so that the creation buffer will be 32 bits aligned

Each array is not necessarily aligned on 32 bits. The alignment is provided by adding zero, one, two or three NULL character(s) at the end of the last array (StringArray).

The specific event sent by the PopUpMenu to the owner application is defined in Table 3.3.31-3.

Table 3.3.31-3 – PopUpMenu Event Structures: A661_EVT_POPUP_CLOSED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_POPUP_CLOSED
SelectedEntry	uchar	8	0 when the pop up is closed without any selection ‘n’ in [1; NumberOfEntries] else.
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.31-4.

Table 3.3.31-4 – PopUpMenu Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
StringArray [NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray
EnableArray [**NumberOfEntries**]	N/A	{32}+	A661_ENABLE_ARRAY	A661_ParameterStructure_EnableArray Refer to definition in table 4.5.4.5.7-1 .
PictureArray [**NumberOfEntries**]	N/A	{32}+	A661_PICTURE_ARRAY	A661_ParameterStructure_PictureArray Refer to definition in table 3.3.31.1-1 below.
StringArray [NumberOfEntries] and EnableArray [NumberOfEntries]	N/A	{32}+	A661_ENTRY_POP_UP_ARRAY	A661_ParameterStructure_EntryPopUp Array Refer to definition in table 3.3.31.1-2 below.

3.3.31.1 PopUp Specific A661_ParameterStructure

A661_ParameterStructure_PictureArray is defined in Table 3.3.31.1-1.

Table 3.3.31.1-1 – A661_ParameterStructure_PictureArray

A661_ParameterStructure_PictureArray	Size (bits)	Description
Parameter_ident	16	A661_PICTURE_ARRAY
EntryIndex	8	
UnusedPad	8	0
Picture	16	Picture reference
UnusedPad	16	0

A661_ParameterStructure_EntryPopUpArray is defined in Table 3.3.31.1-2.

Table 3.3.31.1-2 – A661_ParameterStructure_EntryPopUpArray

A661_ParameterStructure_EntryPopUpArray	Size (bits)	Description
Parameter_ident	16	A661_ENTRY_POP_UP_ARRAY
Number Of Entr**ies** Updated	16	
{ EntryPopUp_Structure }+	{32}+	Refer to table 3.3.31.1-3.

Table 3.3.31.1-3 – EntryPopUp_Structure

EntryPopUp_Structure	Size (bits)	Description
EntryIndex	8	
Enable	8	A661_FALSE A661_TRUE
Picture	16	
StringLength	16	
String	8 * string length + PAD	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned.

3.3.32 PopUpMenuButton

Categories:

- Graphical representation
- **Interactive**
- Text string

Description:
The PopUpButton widget contains a Button widget that displays a PopUpMenu, which is internal to the CDS.

This widget contains a PopUpMenu widget. The UA has the responsibility to define the position of the PopUpMenu.

Restriction:
None

PopUpMenuButton Parameters are defined in Table 3.3.32-1.

Table 3.3.32-1 – PopUpMenuButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_POP_UP_MENU_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters for the button		
MaxStringLength	D	Maximum length of the label text
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT

Parameters	Change	Description
		CENTER
LabelString	DR	Label of the menu button
Picture Reference	DR	Reference of the picture to be displayed on the button
PicturePosition	D	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
Specific Parameters of PopUp		
PopupPosX	D	The X position of the PopUpMenu reference point
PopupPosY	D	The Y position of the PopUpMenu reference point
PopupSizeX	D	The X dimension size (width) of the PopUpMenu
PopupSizeY	D	The Y dimension size (height) of the PopUpMenu
OpeningMode	D	OPEN_UP: the position (X,Y) is given according to bottom/left point OPEN_DOWN: the position (X,Y) is given according to top/left point
NumberOfEntries	D	Number of entries in the PopUpMenu. It also includes the graphical separators.
MaxStringLengthPopUp	D	Maximum string length for the entries on the popup, including the first NULL character ending the string (in bytes).
StringArray	DR	String attached to one entry Strings composed by only one NULL character will be interpreted as a graphical separator. The NULL string at the end of the array will not be interpreted.
PictureArray	DR	Picture attached to each entry (if value is NULL, no picture is attached to the corresponding entry).
PopUpIdent Array	D	ushort for the PopUpMenu attached to one string. WidgetIdent can only refer to another PopUpMenu. If WidgetIdent is NULL, no PopUpMenu is attached to this Entry.
EnableArray	DR	Ability for each Entry on the PopUpMenu: Refer to table 4.5.4.5.7-1.
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

PopUpMenuButton Creation Structure is defined in Table 3.3.32-2.

Table 3.3.32-2 – PopUpMenuButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_POP_UP_MENU_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
PopupPosX	long	32	
PopupPosY	long	32	
PopupSizeX	ulong	32	
PopupSizeY	ulong	32	
MaxStringLength	ushort	16	
MaxStringLengthPopUp	ushort	16	
PictureReference	ushort	16	
NumberOfEntries	uchar	8	
PicturePosition	uchar	8	A661_CENTER A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_DOWN
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned
PopUpIdentArray[NumberOfEntries]	{ushort}+	16 * NumberOf Entries	
PictureArray [NumberOfEntries]	{ushort}+	16 * NumberOfEntries	
EnableArray[NumberOfEntries]	{uchar}+	8 * NumberOf Entries	
StringArray[NumberOfEntries]	{string}+	8 * string length + Pad	There are “NumberOfEntries” strings. Each string is ended by character NULL (part of the string used as string separator). The complete string list is followed by zero, one, two or three NULL character(s) so that the creation buffer will be 32 bits aligned

Each array is not necessary aligned on 32 bits. The alignment is provided by adding zero, one, two or three NULL character(s) at the end of the last array only (StringArray)

The specific event sent by the PopUpMenuButton to the owner application is defined in Table 3.3.32-3.

Table 3.3.32-3 – PopUpMenuButton Event Structures: A661_EVT_POPUP_CLOSED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_POPUP_CLOSED
SelectedEntry	uchar	8	0 when the pop up is closed without any selection else 'n' in [1; NumberOfEntries].
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.32-4.

Table 3.3.32-4 – PopUpMenuButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
PictureReference	ushort	16	A661_PICTURE_REFERENCE	A661_ParameterStructure_2Bytes
StringArray [NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray
EnableArray [Entries]	N/A	{32}+	A661_ENABLE_ARRAY	A661_ParameterStructure_EnableArray Refer to definition in Table 4.5.4.5.7-1
PictureArray [Entries]	N/A	{32}+	A661_PICTURE_ARRAY	A661_ParameterStructure_PictureArray Refer to definition in table 3.3.31.1-1.
StringArray [NumberOfEntries] And EnableArray [NumberOfEntries]	N/A	{32}+	A661_ENTRY_POP_UP_ARRAY	A661_ParameterStructure_EntryPopUpArray Refer to definition in Table 3.3.31.1-2
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte

3.3.33 PushButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
A PushButton widget is a momentary switched Button, which enables the crew to launch an action.

A PushButton has only one inner state, so there is no need for an inner state parameter.

Restriction:
None

PushButton Parameters are defined in Table 3.3.33-1.

Table 3.3.33-1 – PushButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PUSH_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelString	DR	String of the PushButton
MaxStringLength	D	Maximum length of the label text
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

PushButton Creation Structure is defined in Table 3.3.33-2.

Table 3.3.33-2 – PushButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PUSH_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned

This event indicates to the UA that a crew member has interacted with the widget.

PushButton Event Structures: A661_EVT_SELECTION is defined in Table 3.3.33-3.

Table 3.3.33-3 – PushButton Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	**ushort**	16	A661_EVT_SELECTION
UnusedPad	**N/A**	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.33-4.

Table 3.3.33-4 – PushButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.34 RadioBox

Categories:
Container

Description:
A RadioBox widget manages the visibility and the interactivity of a group of Buttons (CheckButtons or ToggleButtons). It enables a crew member to select one Button out of “n” exclusive ones. At a given time one item maximum can be SELECTED. A selection of a selected item by a crew member is without effect. Nevertheless, the UA can deselect the selected item (through setParameter command) to create a RadioBox without selection. The Buttons contained in the RadioBox should be individually defined with the RadioBox as a parent widget. RadioBox does not have any graphical representation.

Restriction:
The children of the RadioBox will be positioned relative to the parent of the RadioBox. A RadioBox has only children types:

1. ToggleButton
2. PictureToggleButton
3. CheckButton

Only one type can be used in a given RadioBox at a time. The CDS assures that internal state of the children is consistent (no more than one is selected) at all times, including when the user changes the state of the children.

RadioBox Parameters are defined in Table 3.3.34-1.

Table 3.3.34-1 – RadioBox Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_RADIO_BOX
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated

RadioBox Creation Structure is defined in Table 3.3.34-2.

Table 3.3.34-2 – RadioBox Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_RADIO_BOX
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE

The RadioBox widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.34-3.

Table 3.3.34-3 – RadioBox Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte

3.3.35 RotationContainer

Categories:
Container

Description:
A RotationContainer widget applies a rotation transformation to a group of widgets. Widgets placed within RotationContainer have their coordinates referenced to the first parent with a PosX, PosY reference point.

Restriction:
For RotationContainer restriction refer to Table 3.2.3.1 for children/parents.

RotationContainerParameters are defined in Table 3.3.35-1.

Table 3.3.35-1 – RotationContainerParameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ROTATION_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Specific parameters		
CenterX	DR	X position of the center of the rotation
CenterY	DR	Y position of the center of the rotation
RotationAngle	DR	Rotation angle to be applied to the children widgets

RotationContainer Creation Structure is defined in Table 3.3.35-2.

Table 3.3.35-2 – RotationContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ROTATION_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
CenterX	long	32	
CenterY	long	32	
RotationAngle	fr(180)	32	

The RotationContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.35-3.

Table 3.3.35-3 – RotationContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
CenterX CenterY	long x 2	32 x 2	A661_CENTER_XY	A661_ParameterStructure_8Bytes
CenterX	long	32	A661_CENTER_X	A661_ParameterStructure_4Bytes
CenterY	long	32	A661_CENTER_Y	A661_ParameterStructure_4Bytes
RotationAngle	fr(180)	32	A661_ROTATION_ANGLE	A661_ParameterStructure_4Bytes

3.3.36 ScrollPanel

Categories:

- Container
- Graphical representation

Description:
A scroll container is composed of two elements:

Frame, at fixed location. This location is the position of the widget (as already known), defined by the parameters PosX, PosY, SizeX, SizeY.

Sheet, larger than the Frame, at a variable location with respect to the Frame. This location is defined by the variables FrameX, FrameY, SizeXsheet, SizeYsheet. Note that X/Y coordinates of the sheet are called FrameX and FrameY.

Indeed, the sheet X/Y coordinates should in fact be interpreted as the offset of the sheet relative to the frame according to standard coordinate system, shown in Figure 3.3.36.

The scrolling function is allowed by DeltaX, DeltaY parameters, which provide to the CDS the displacement of the sheet to apply when a crew member initiates an action

with the scroll controls. The type of scroll controls provided are CDS OEM dependent.

The scrolling function is also subject to boundaries specified through BoundX, BoundY, SizeXbound, SizeYbound parameters accessible by the UA at run time. These coordinates refer to the sheet location.

The CDS should provide scroll controls (scroll bars and/or scroll buttons, according to the airframe manufacturer/system integrator style guide). Typically, this is based on the relative size of the frame and the sheet. For instance, if X size of the frame is smaller than the X size of the sheet, the CDS should set a horizontal scroll control. Two parameters are available to allow the UA to choose from a variety of positions according to the airframe manufacturer/system integrator style guide.

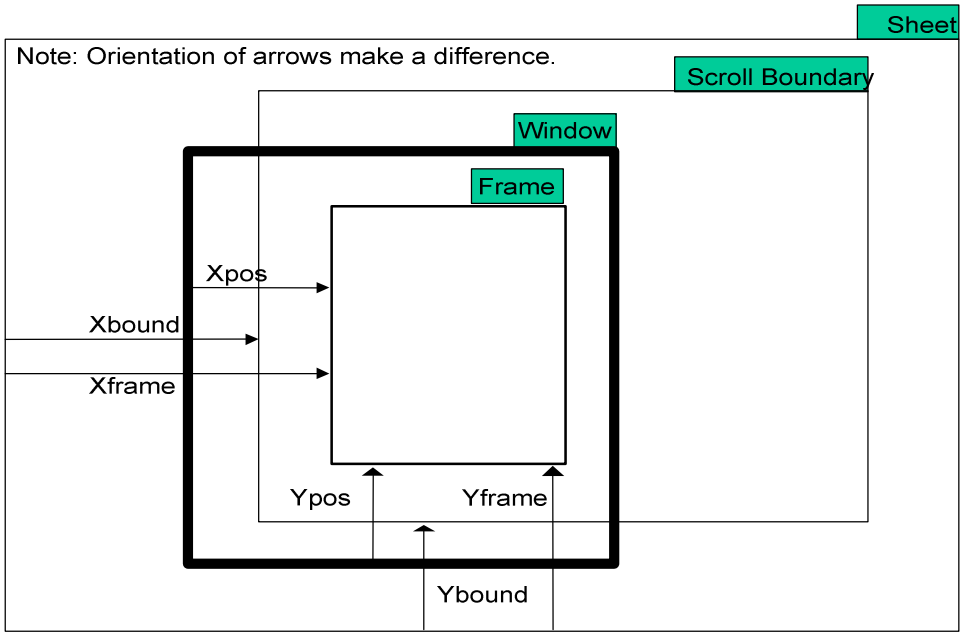


Figure 3.3.36 – Frame Standard Coordinate System

Restriction: The reference position for the children of the ScrollPanel is the FrameX and FrameY.

ScrollPanel Parameters are defined in Table 3.3.36-1.

Table 3.3.36-1 – ScrollPanel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SCROLL_PANEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Style Set	DR	Reference to predefined graphical characteristics inside CDS
Enable	DR	Ability of the widget to be activated
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
LineDeltaX	D	Increment/Decrement to apply to FrameX when line scroll controls are activated.
LineDeltaY	D	Increment/Decrement to apply to FrameY when line scroll controls are activated.
PageDeltaX	D	Increment/Decrement to apply to FrameX when page scroll controls are activated.
PageDeltaY	D	Increment/Decrement to apply to FrameY when page scroll controls are activated.
HomeX	D	X predefined position for the frame
HomeY	D	Y predefined position for the frame
FrameX	DR	Frame Origin co-ordinate on x axis.
FrameY	DR	Frame Origin co-ordinate on y axis.
SizeXsheet	D	X dimension size of the sheet
SizeYsheet	D	Y dimension size of the sheet
BoundX	DR	Scroll Boundary Origin co-ordinate on x axis.
BoundY	DR	Scroll Boundary Origin co-ordinate on y axis.
SizeXbound	DR	X dimension size of the Scroll boundary
SizeYbound	DR	Y dimension size of the Scroll boundary
FlagReportFramePos	D	If True, CDS will report change on the frame position following crew member actions.
Horizontal Scroll	D	Absent/Top/Bottom/Left/Right
Vertical Scroll	D	Absent/Left/Right/Top/Bottom
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

ScrollPanel Creation Structure is defined in Table 3.3.36-2.

Table 3.3.36-2 – ScrollPanel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SCROLL_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
LineDeltaX	ulong	32	
LineDeltaY	ulong	32	
PageDeltaX	ulong	32	
PageDeltaY	ulong	32	
HomeX	long	32	
HomeY	long	32	
FrameX	long	32	
FrameY	long	32	
SizeXsheet	ulong	32	
SizeYsheet	ulong	32	
BoundX	long	32	
BoundY	long	32	
SizeXbound	ulong	32	
SizeYbound	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0
Horizontal Scroll	uchar	8	A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT A661_ABSENT
Vertical Scroll	uchar	8	A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT A661_ABSENT
FlagReportFramePos	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0

ScrollPanel Event Structures: A661_EVT_FRAME_POS_CHANGE are defined in Table 3.3.36-3.

Table 3.3.36-3 – ScrollPanel Event Structures: A661_EVT_FRAME_POS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_FRAME_POS_CHANGE
UnusedPad	ushort	16	0
FrameX	long	32	
FrameY	long	32	

Available SetParameter identifiers and associated data structure are defined in Table 3.3.36-4.

Table 3.3.36-4 – ScrollPanel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
FrameX	long	32	A661_FRAME_X	A661_ParameterStructure_4Bytes
FrameY	long	32	A661_FRAME_Y	A661_ParameterStructure_4Bytes
FrameX FrameY	long x 2	32 x 2	A661_FRAME_XY	A661_ParameterStructure_8Bytes
BoundX	long	32	A661_BOUND_X	A661_ParameterStructure_4Bytes
BoundY	long	32	A661_BOUND_Y	A661_ParameterStructure_4Bytes
BoundX BoundY	long x 2	32 x 2	A661_BOUND_XY	A661_ParameterStructure_8Bytes
SizeXbound	ulong	32	A661_BOUND_SIZE_X	A661_ParameterStructure_4Bytes
SizeYbound	ulong	32	A661_BOUND_SIZE_Y	A661_ParameterStructure_4Bytes
SizeXbound SizeYbound	ulong x 2	32 x 2	A661_BOUND_SIZE_XY	A661_ParameterStructure_8Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.37 ScrollList

Categories:

- Graphical Representation
- Interactive
- Text string

Description:
A ScrollList widget enables the display of a list of entries and selection of one entry from among this list. Entries are text strings, possibly including escape sequences. This is specified through the DefaultStyleText Definition Time Only parameter, if set to null, all labels can be considered by the CDS as being like normal Labels. As a consequence of the use of escape sequences, one entry in the ScrollList can correspond to several lines.

Scroll controls are provided by the CDS. For example, these controls could allow for scrolling by single items or by page. The type of scroll controls provided are CDS OEM dependent.

The FirstVisibleEntry parameter is generally only managed by the CDS and is used to define which LabelStringArray/EnableArray entry is positioned as the first entry in the visible area of the ScrollArray. Although the UA can update this parameter, doing so during normal operation can cause a race condition (as described in Section 3.1.2.2). If the UA needs to update the list of accessible entries such that the LabelStringArray entry containing the currently FirstVisibleEntry has moved, the UA should use the parameter ShiftFirstVisibleEntry to perform a one-time adjustment of the FirstVisibleEntry parameter. The usage of ShiftFirstVisibleEntry is shown in Figure 3.3.37-1.

With the ScrollList widget the UA can maintain a large list of items external to CDS and provide a subset to the ScrollList widget. The subset managed by the ScrollList **(1 through MaxNumberOfEntries)** includes the items that are visible and can also include data within the immediate vicinity of the visible area to provide for rapid scrolling. **The UA uses FirstAccessibleEntry and NumberOfEntries to specify which subset of LabelStringArray and EnableArray contains accessible entries (those which can be made visible by the CDS). Entries not in the range specified by FirstAccessibleEntry and NumberOfEntries (for example, stale data) are not allowed to become visible. An illustration of an update to the accessible entries by the UA is shown in Figure 3.3.37-2.**

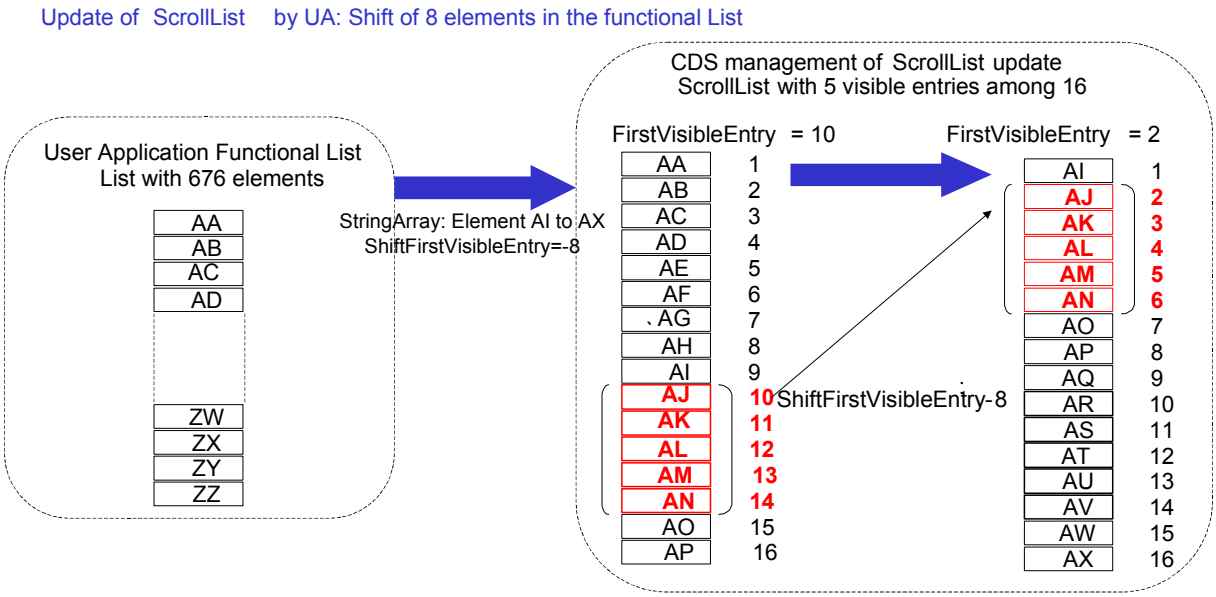


Figure 3.3.37-1 – Scroll List update using ShiftFirstVisibleEntry and LabelStringArray

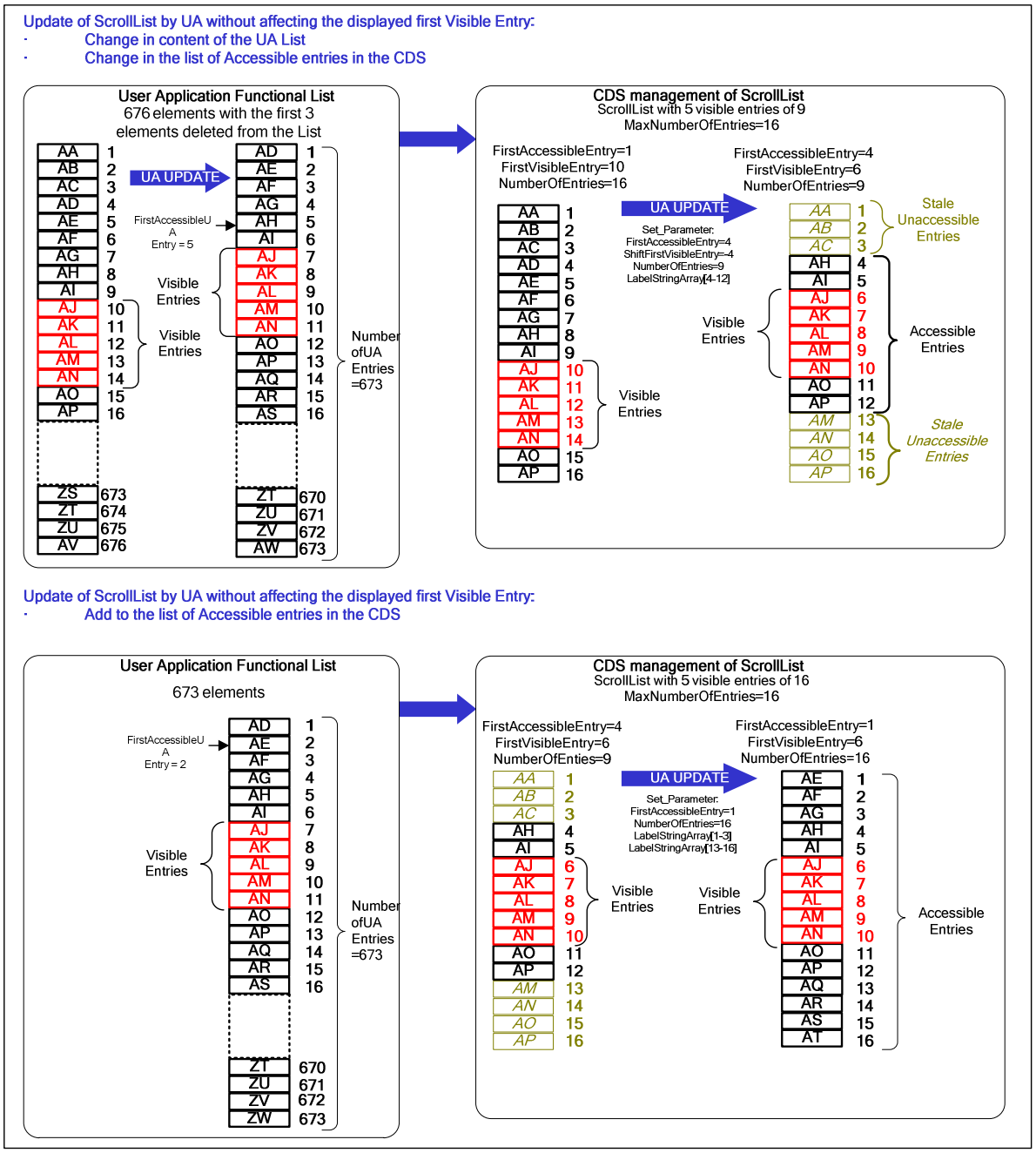


Figure 3.3.37-2 – Scroll List update using ShiftFirstVisibleEntry and FirstAccessibleEntry

Restriction:
SelectedEntry, FirstVisibleEntry and FirstAccessibleEntry assume the first Entry index to be 1. If SelectedEntry is 0, it is interpreted as none.

ScrollList Parameters are defined in Table 3.3.37-1.

Table 3.3.37-1 – ScrollList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SCROLL_LIST
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
NumberOfEntries	DR	Number of accessible entries
MaxNumberOfEntries	D	**Max number of accessible entries to be managed by the CDS**
FirstVisibleEntry	DR	Index **into LabelStringArray/EnableArray of the entry appearing as the first entry in the visible area of the ScrollList**
ShiftFirstVisibleEntry	R	**Index shift to be applied to the current FirstVisibleEntry value. That is, when a UA wants to add or delete entries above the FirstVisibleEntry, this parameter allows the UA to cause the CDS to change the value of FirstVisibleEntry by the number of deleted/inserted entries. FirstVisibleEntry is updated each time the ShirtFirstVisibleEntry parameter is received.**
FirstAccessibleEntry	DR	**The Entry number of the first entry in the LabelStringArray/EnableArray which the CDS manages as part of the scrollable items in the ScrollList.**
NumberOfUAEntries	**R**	**Number of entries in the UA managed list of Items.**
FirstAccessibleUAEntry	**R**	**Index into the UA managed list of items that corresponds to the FirstAccessibleEntry parameter. This parameter combined with NumberOfUAEntries allows the CDS to display the positioning of the visible entries in the ScrollList relative to the UA managed list. Note that not all CDS implementations display this information and may ignore these parameters.**
FlagReportVisibleEntry	D	If True, CDS will report change on first visible entry following crew member actions.
SelectedEntry	DR	Currently selected Entry index

Parameters	Change	Description
DefaultStyleText	D	NULL character: Escape sequence not used, entries in the ScrollList are simple labels. "ToutLine⊗TBackColor⊗TForeColor⊗TFont"
MaxStringLength	D	Maximum string length able to be received by the object including the first NULL character ending the string (in bytes), MaxStringLength > 1 (at least one significant character).
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelStringArray[MaxNumberOfEntries]	DR	Label array of the ScrollList
EnableArray[MaxNumberOfEntries]	DR	Ability for each Entry on the **ScrollList to be selected** : TRUE FALSE
Vertical Scroll	D	ABSENT LEFT RIGHT TOP BOTTOM
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

ScrollListCreation Structure is defined in Table 3.3.37-2.

Table 3.3.37-2 – ScrollList Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SCROLL_LIST
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
FirstAccessibleEntry	ushort	16	
FirstVisibleEntry	ushort	16	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
Vertical Scroll	uchar	8	**A661_TOP** **A661_BOTTOM** A661_LEFT A661_RIGHT A661_ABSENT
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
FlagReportVisibleEntry	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
DefaultStyleText	uchar	96	
EnableArray [NumberOfEntries]	{uchar} +	8 * NumberOf Entries	Enable status of 'NumberOfEntries' Entries from FirstAccessibleEntry
LabelStringArray	{string}+	8 * string lengths + Pad	There are "NumberOfEntries" strings. Each string terminating NULL is used as string separator. The complete string list is followed by zero, one, two or three NULL character(s) to be 32 bits aligned.

ScrollList Event Structures: A661_EVT_SEL_ENTRY_CHANGE are defined Table 3.3.37-3.

Table 3.3.37-3 – ScrollList Event Structures: A661_EVT_SEL_ENTRY_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	**ushort**	16	A661_EVT_SEL_ENTRY_CHANGE
SelectedEntry	**ushort**	16	Index of the new selected entry

ScrollList Event Structures: A661_EVT_FIRST_VIS_ENTRY_CHANGE are defined in Table 3.3.37-4.

Table 3.3.37-4 – ScrollList Event Structures: A661_EVT_FIRST_VIS_ENTRY_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	**ushort**	16	A661_EVT_FIRST_VIS_ENTRY_CHANGE
FirstVisibleEntry	**ushort**	16	Index of the first visible entry

Available SetParameter identifiers and associated data structure are defined in Table 3.3.37-5.

Table 3.3.37-5 – ScrollList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES	A661_ParameterStructure_2Bytes
NumberOfUAEntries	**ushort**	**16**	**A661_NUMBER_OF_UA_ENTRIES**	**A661_ParameterStructure_2Bytes**
FirstAccessibleEntry	ushort	16	A661_FIRST_ACCESS_ENTRY	A661_ParameterStructure_2Bytes
FirstAccessibleUAEntry	**ushort**	**16**	**A661_FIRST_ACCESS_UA_ENTRY**	**A661_ParameterStructure_2Bytes**
FirstVisibleEntry	ushort	16	A661_FIRST_VISIBLE_ENTRY	A661_ParameterStructure_2Bytes
ShiftFirstVisibleEntry	short	16	A661_SHIFT_FIRST_VISIBLE_ENTRY	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	16	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
LabelStringArray[Max NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray
EnableArray [Entry]	N/A	{32}+	A661_ENABLE_ARRAY	A661_ParameterStructure_EnableArray Refer to Table 4.5.4.5.7-1
LabelStringArray [MaxNumberOfEntries] and EnableArray [MaxNumberOfEntries]	N/A	{32}+	A661_ENTRY_ARRAY	A661_ParameterStructure_EntryArray Refer to Definition Below
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.37.1 ScrollList Specific A661_ParameterStructure

A661_ParameterStructure_EntryArray is defined in Table 3.3.37.1-1.

Table 3.3.37.1-1 – A661_ParameterStructure_EntryArray

A661_ParameterStructure_Entry Array	Size (bits)	Description
Parameter_ident	16	A661_ENTRY_ARRAY
Number Of Entries Updated	16	
{ EntryScrollList_Structure }+		Reference table 3.3.37.1-2

Table 3.3.37.1-2 – EntryScrollList_Structure

EntryScrollList_Structure	Size (bits)	Description
StringLength	16	
EntryIndex	16	
Enable	8	A661_FALSE A661_TRUE
UnusedPad	8	0
String	8 * string length + PAD	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned

3.3.38 Symbol

Categories:

- Graphical Representation
- Dynamic Motion

Description:
The Symbol widget is similar to the Label widget, except it does not have a Max-String-Length parameter and the string parameter is replaced by a Symbol-Reference parameter (outside reference).

Restriction:
None

Symbol Parameters are defined in Table 3.3.38-1.

Table 3.3.38-1 – Symbol Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SYMBOL
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
Specific parameters		
MotionAllowed	D	Capability to change PosX, PosY, Rotation Angle at runtime
RotationAngle	DR	Angle at which symbol is displayed relative to its origin Refer to Angles defined in Section 2.3.4.2
ColorIndex	DR	Color index of the symbol, used if StyleSet allows color to be set.
SymbolReference	DR	Reference of the symbol stored in the CDS

Symbol Creation Structure is defined in Table 3.3.38-2.

Table 3.3.38-2 – Symbol Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SYMBOL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Motion Allowed	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
RotationAngle	fr(180)	32	
StyleSet	ushort	16	
SymbolReference	ushort	16	
ColorIndex	uchar	8	(valid palette index)
UnusedPad	N/A	24	0

Symbol does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.38-3.

Table 3.3.38-3 – Symbol Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
RotationAngle	fr(180)	32	A661_ORIENTATION	A661_ParameterStructure_4Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_1Byte
SymbolReference	ushort	16	A661_SYMBOL_REFERENCE	A661_ParameterStructure_2Bytes

3.3.39 TabbedPanel

Categories:

- Container
- Graphical Representation
- Text string

Description:
The TabbedPanel widget is functionally composed of a Panel associated with a Button. This widget can be created only inside a TabbedPanelGroup widget. The size of the panel part of the TabbedPanel widget is identical for all the TabbedPanels inside a TabbedPanelGroup and is therefore described by the TabbedPanelGroup widget. Connectors can be used to move the definition of the TabbedPanel to a different definition file so that the owning application can control the parameters of the TabbedPanel.

The TabbedPanel widget is not interactive, however it contains a **Next**Focus**edWidget** parameter used by the parent TabbedPanelGroup to shift focus from one tab to another.

Restriction:
The TabbedPanel widget should only be used under a TabbedPanelGroup or a Layer. When directly attached to a layer, this layer should not be attached to a window to be displayed alone.

TabbedPanel Parameters are defined in Table 3.3.39-1.

Table 3.3.39-1 – TabbedPanel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TABBED_PANEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to graphical characteristics defined inside CDS. The StyleSet will influence only the label or picture displayed on the button associated with the TabbedPanel
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
LabelString	DR	Label of the tab
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
MaxStringLength	D	Maximum string length of the label
InsetSize	D	Size of the button associated with the TabbedPanel, in the direction of the text writing, in screen units (millimeters).
Picture Reference	DR	Picture reference among available picture inside CDS
PicturePosition	D	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM

COMMENTARY

TabbedPanel and TabbedPanelGroup widgets are defined as separate widgets to provide the UA the ability to change the characteristics of each TabbedPanel when it is necessary. This implies that there will be one identifier for the TabbedPanelGroup and one identifier per TabbedPanel child **widget**.

TabbedPanel Creation Structures are defined in Table 3.3.39-2.

Table 3.3.39-2 – TabbedPanel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TABBED_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
PictureReference	ushort	16	
PicturePosition	uchar	8	A661_CENTER A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
UnusedPad	N/A	8	0
InsetSize	ulong	32	
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

The TabbedPanel widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.39-3.

Table 3.3.39-3 – TabbedPanel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
PictureReference	ushort	16	A661_PICTURE_REFERENCE	A661_ParameterStructure_2Bytes

3.3.40 TabbedPanelGroup

Categories:

- Container
- Graphical representation
- Interactive

Description:
A TabbedPanelGroup widget groups several TabbedPanel widgets. A TabbedPanelGroup enables the UA or a crew member to select one of the TabbedPanel widgets for display. All of the **Tabbed**Panels inside the TabbedPanel**Group** widget occupy the same display space, and only one may be displayed at a time. Th**e displayed** TabbedPanel is the one referenced by the "Active TabbedPanel ID" **parameter**.

The TabbedPanelGroup has clipping capabilities.

Restriction:
A TabbedPanelGroup can only contain TabbedPanel or Connector widgets.

TabbedPanel Group Parameters are defined in Table 3.3.40-1.

Table 3.3.40-1 – TabbedPanelGroup Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TABBED_PANEL_GROUP
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
TabPosition	D	Display and position of optional tab: ABSENT no tab will be used TOP automatic tab will be set up BOTTOM automatic tab will be set down LEFT automatic tab will be set left RIGHT automatic tab will be set right
AutomaticInsetSizeFlag	D	If TabPosition is Top/Bottom: TRUE: CDS defines the button size according to the TabbedPanelGroup and the number of button. FALSE: The button size is defined by the TabbedPanel parameter **B**uttonSize. If this size is incoherent, it is set by the CDS automatically If TabPosition is Right/Left: TRUE: CDS defines the button size according to the StyleSet FALSE: The CDS use the maximum of the sizes defined

Parameters	Change	Description
		inside the TabbedPanel
ActiveTabbedPanelID	DR	Identifier of the active TabbedPanel **or the associated Connector reference**
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

TabbedPanelGroup Creation Structure is defined in Table 3.3.40-2.

Table 3.3.40-2 – TabbedPanelGroup Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TABBED_PANEL_GROUP
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
ActiveTabbedPanelID	ushort	16	
TabPosition	uchar	8	A661_ABSENT A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT
AutomaticInsetSizeFlag	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

TabbedPanelGroup Event Structures: A661_EVT_TABBED_PANEL_CHANGE are defined in Table 3.3.40-3.

Table 3.3.40-3 – TabbedPanelGroup Event Structures: A661_EVT_TABBED_PANEL_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_TABBED_PANEL_CHANGE
ActiveTabbedPanelID	ushort	16	Identifier of the selected TabbedPanel **or the associated Connector reference**

Available SetParameter identifiers and associated data structure are defined in Table 3.3.40-4.

Table 3.3.40-4 – TabbedPanelGroup Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
ActiveTabbedPanelID	ushort	16	A661_ACTIVE_TABBED_PANEL	A661_ParameterStructure_2Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.41 ToggleButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
A ToggleButton widget is a two, stable-states Button with text.

Restriction:
None

ToggleButton Parameters are defined in Table 3.3.41-1.

Table 3.3.41-1 – ToggleButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TOGGLE_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
ToggleState	DR	Inner state of the ToggleButton UNSELECTED SELECTED
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	Maximum length of the label text
AlternateFlag	D	True: Use of the two strings according to the inner state. CDS will change the string if the inner state change False: “AlternateString” is not used. Only parameter “string” is used for the two inner state
Alignment	D	Alignment of the text within the label area of the widget

Parameters	Change	Description
		LEFT RIGHT CENTER
String	DR	Label of the ToggleButton Label used for UNSELECTED state
AlternateString	DR	Label of the ToggleButton Label used for SELECTED state
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

ToggleButton Creation Structure is defined in Table 3.3.41-2.

Table 3.3.41-2 – ToggleButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TOGGLE_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
InnerState	uchar	8	A661_UNSELECTED A661_SELECTED
AlternateFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
UnusedPad	N/A	16	0
LabelString	string	8 * string length	String terminator NULL is used as string separator.
AlternateLabelString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

ToggleButton Event Structures: A661_EVT_STATE_CHANGE are defined in Table 3.3.41-3.

Table 3.3.41-3 – ToggleButton Event Structures: A661_EVT_STATE_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STATE_CHANGE
InnerState	uchar	8	A661_UNSELECTED A661_SELECTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.3.41-4.

Table 3.3.41-4 – ToggleButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
ToggleState	uchar	8	A661_INNER_STATE_TOGGLE	A661_ParameterStructure_1Byte
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
AlternateLabelString	string	{32}+	A661_STRING_ALTERNATE	A661_ParameterStructure_String
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.3.42 TranslationContainer

Categories:
Container

Description:
A TranslationContainer widget applies a translation transformation to a group of widgets. Widgets placed within TranslationContainer have their coordinates referenced to the first parent with a PosX, PosY reference point.

Restriction:
For TranslationContainer restriction refer to Table 3.2.3.1 regarding children/parents.

TranslationContainer Parameters are defined in Table 3.3.42-1.

Table 3.3.42-1 – TranslationContainerParameters Table

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TRANSLATION_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Specific parameters		
TranslationX	DR	X Translation of the child widgets
TranslationY	DR	Y Translation of the child widgets

TranslationContainer Creation Structure is defined in Table 3.3.42-2.

Table 3.3.42-2 – TranslationContainer Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TRANSLATION_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
TranslationX	long	32	
TranslationY	long	32	

The TranslationContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.42-3.

Table 3.3.42-3 – TranslationContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
TranslationX TranslationY	long x 2	32 x 2	A661_TRANSLATION_XY	A661_ParameterStructure_8Bytes
TranslationX	long	32	A661_TRANSLATION_X	A661_ParameterStructure_4Bytes
TranslationY	long	32	A661_TRANSLATION_Y	A661_ParameterStructure_4Bytes

3.4 Widget Library Expansion

This section was added in Supplement 1. It introduces new widgets to ARINC 661.

3.4.1 MapGrid

Categories:

- Map Management
- Graphical Representation

Description:
MapGrid provides a means for conveying arrays of data to the CDS that are rendered as area fills. The intended use is for filling areas on background layers of the NAV window with colors and/or patterns that indicate terrain topography, precipitation intensity, or other irregular, dynamic data.

The fill is defined by the number of cells in the horizontal and vertical, the size of each cell in nautical miles or equivalent, the offset of the grid's (0,0) cell from the display origin, and the Fill Style Index for each cell. Distances are described in real-world units, which decouples the UA from the specific display technology. The entire area defined by each cell boundary is to be filled with the color or pattern or other graphical attribute as selected by the Fill Style Index. Typically, slightly more data is supplied than is displayed. The amount of excess depends on several factors:

- If the CDS or UA implements motion compensation (update the origin or rotation independently of color data)
- If the background data is masked around the edges
- If the application is aware of the current display mode (arc, rose, plan, center, etc.)
- If there is sufficient bandwidth between UA and CDS for an oversized array
- If there is sufficient memory allocated in the CDS for an oversized array

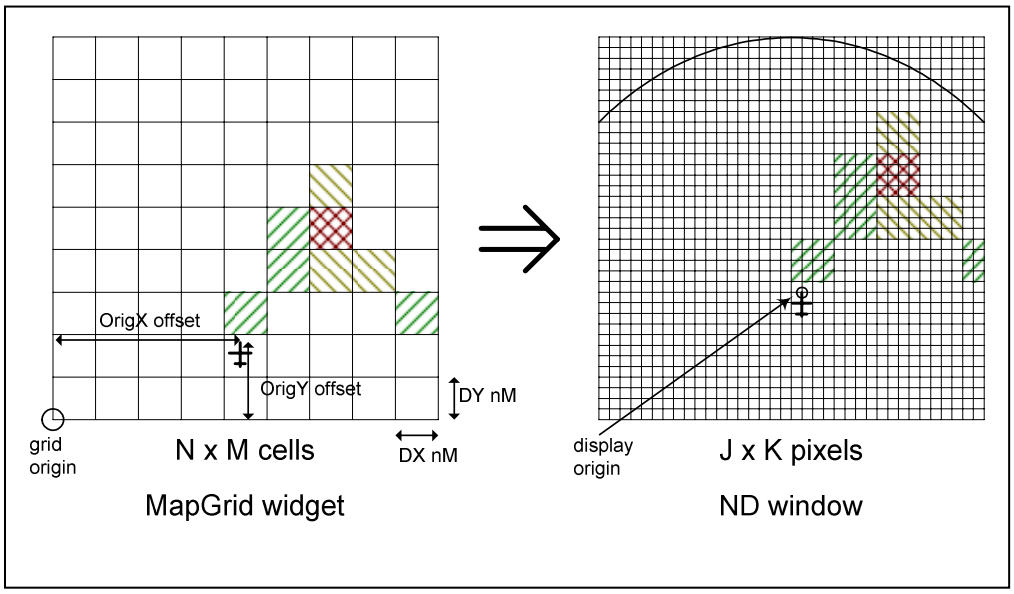


Figure 3.4.1-1 – Example MapGrid rendering in ND window

The UA may need to update the MapGrid color data periodically. Since the array may be large relative to the bandwidth available, provision is made for just a few (or one) rows or columns at a time. The UA can change the size of a cell, in real-world units, at run-time to support a balance between range and resolution. The size of a MapGrid array, in cells, is fixed at Definition Time.

Restriction:
A MapGrid must be in a MapHorz_Source or MapVert_Source container.
Support for the various MapData Format Values (table 3.3.24-2b) and MapVert_MapData Format Values (table 3.4.4-2b) depends on the implementation.

MapGrid Parameters are defined in Table 3.4.1-1.

Table 3.4.1-1 – MapGrid Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAP_GRID
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Specific parameters		
CountX	D	Number of cells along the X axis in the array.
CountY	D	Number of cells along the Y axis in the array.
OffsetX	DR	Horizontal offset, in (fractional) cells, between the display reference point and the grid origin. If translation/rotation is done by the UA, this number is constant (typically one-half of CountX). (OffsetX, OffsetY) defines the point in the grid to be placed at the display origin (typically, the aircraft current location). The point may or may not be at a cell boundary, depending on the ratio of cell size to pixel size, chosen aircraft mock-up location, and whether translation for aircraft motion is implemented in the CDS or the UA.
OffsetY	DR	Vertical offset, in (fractional) cells, between the display reference point and the grid origin.
IncrementX	DR	Size of each individual cell in the X axis, in the real-world units defined by the containing Map Source. Support for the various Map Source Data Formats depends on the implementation. MapHorz_Source: A661_MDF_LAT_LONG: See table 3.3.24-2b. A661_MDF_DIST_DIST: See table 3.3.24-2b. A661_MDF_BRG_DIST_ACHDG: See table 3.3.24-2b. A661_MDF_LEGACY See table 3.3.24-2b. MapVert_Source: A661_MDF_X_DIST See table 3.4.4-2b A661_MDF_RELATIVE See table 3.4.4-2b A661_MDF_ABSOLUTE See table 3.4.4-2b
IncrementY	DR	Size of each individual cell in the Y axis, in the real-world units defined by the containing Map Source. Support for the various Map Source Data Formats depends on the implementation. MapHorz_Source: A661_MDF_LAT_LONG: See table 3.3.24-2b. A661_MDF_DIST_DIST: See table 3.3.24-2b. A661_MDF_BRG_DIST_ACHDG: See table 3.3.24-2b. A661_MDF_LEGACY See table 3.3.24-2b. MapVert_Source: A661_MDF_Y_ALT See table 3.4.4-2b. A661_MDF_RELATIVE See table 3.4.4-2b A661_MDF_ABSOLUTE See table 3.4.4-2b
BufferOfFillStyles	R	Buffer of Fill Style Indices. Buffer can be updated row-at-a-time or column-at-a-time.
MapSynchronizationNumber	**R**	**See section 3.2.8.4**

MapGrid Creation Structure is defined in Table 3.4.1-2.

Table 3.4.1-2 – MapGrid Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_MAP_GRID
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	
OffsetX	float	32	range is 0 to CountX
OffsetY	float	32	range is 0 to CountY
IncrementX	float	32	units defined by containing MapHorz_Source widget (degrees or nautical miles)
IncrementY	float	32	units defined by containing MapHorz_Source widget (degrees, nautical miles, or feet)
CountX	ushort	16	
CountY	ushort	16	

MapGrid Runtime Modifiable Parameters are defined in Table 3.4.1-3.

Table 3.4.1-3 – MapGrid Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to sections 4.5.3)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
OffsetX, OffsetY	float x 2	32 x 2	A661_MAPGRID_OFFSET	A661_ParameterStructure_ 8Bytes
IncrementX, IncrementY	**float** x 2	32 x 2	A661_MAPGRID_CELLSIZE	A661_ParameterStructure_ 8Bytes
BufferOfFillStyles	N/A	{32}+	A661_BUFFER_OF_FILL_STYLES	A661_ParameterStructure_BufferOfFillStyles Refer to “MapGrid Parameter Structure Specifics” section below.
MapSynchronizationNumber	**ushort**	**16**	**A661_MAP_SYNCHRONIZATION_NUMBER**	**A661_ParameterStructure_2Bytes See section 3.2.8.4**

All the data in the buffer must use the same origin/orientation values. Even though the buffer can be filled line-at-a-time, over time, all the lines are displayed simultaneously, and must be internally consistent.

For MapGrids in MapHorz_Sources that have a variable orientation value (A661_MDF_BRG_DIST_ACHDG or A661_MDF_DIST_DIST), the convention is that the first line of fill (the one following a BUFFER_COMPLETE signal – see next section), must be aligned to the current orientation value (aircraft heading). If the orientation reference changes during subsequent line updates, those subsequent lines must be oriented consistent with all the preceding ones (in particular, the first one).

3.4.1.1 MapGrid A661_ParameterStructure Specifics

Table 3.4.1.1-1 – A661_ParameterStructure_BufferOfFillStyles

Field	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
StartIndexX	ushort	16	Index (zero-based) of column to start storing data.
StartIndexY	ushort	16	Index (zero-based) of row to start storing data. Some CDS implementations may require that one of StartIndexX or StartIndexY be zero.
NumColumns	ushort	16	Number of columns being filled.
NumRows	ushort	16	Number of rows being filled. Some CDS implementations may require that one of NumColumns or NumRows be set to CountX or CountY, respectively. For some implementations the restriction may only be enforced if the other of NumColumns, NumRows is greater than one.
StepX	uchar	8	+1 or –1 (0xFF) May be set to 0 if NumColumns is 1.
StepY	uchar	8	+1 or –1 (0xFF) May be set to 0 if NumRows is 1. Some CDS implementations may suggest or require that StepX and/or StepY always be set to a specific value (such as +1).
RowMajor	uchar	8	A661_FALSE or A661_TRUE If TRUE, the second Fill Style Index in this message goes into the same COLUMN as the first. If FALSE, the second one goes into the same ROW as the first. Some CDS implementations may suggest or require this parameter always be set to a given value.
ControlFlag	uchar	8	bit 0 = clear buffer (see text) bit 1 = buffer complete (see text)
ParameterValue	uchar	{32}	List of Fill Style Index values. Data are stored starting with the cell indexed by [StartIndexX, StartIndexY] and continuing in the direction specified by the RowMajor parameter until the specified NumColumns (or NumRows) have been filled, then moving one row (or column) in the direction specified by the StepX or StepY parameter and repeating until the specified NumRows (or NumColumns) have been filled. Structure is ended by zero, one, two, or three NULL character(s) to pad the structure to 32-bit alignment.

StepX, StepY, and RowMajor typically are constants chosen to work efficiently with the hardware. By listing them specifically in the message, sender and receiver communicate and check their assumptions. If a CDS implementation has restrictions on StartIndexX/Y or NumColumns/NumRows or StepX/StepY/RowMajor values as noted in the descriptions above, and a UA violates those restrictions, the CDS must return an Error Notification Structure (see Section 4.4.2)

The ControlFlag parameter serves two purposes. In the normal case, it must be set to zero. When the least significant bit is set, it indicates the entire buffer should be cleared to a Fill Style Index of zero BEFORE this line of data is stored. This allows the UA to blank the display quickly when required. The meaning of Fill Style Index zero is not defined here (may be all black, all white, or something else, depending on flight deck design).

When ControlFlag bit 1 is set, this indicates that the buffer update will be “complete” AFTER this line of data is stored. This may have a variety of effects. For example, if double buffering is implemented, it allows the CDS to know when to swap buffers. Or, if motion compensation is implemented, it allows the CDS to know that a new frame of data aligned to the current orientation is ready to be sent. User applications should set this flag whenever this sort of action would be appropriate.

The initial state of the buffer before any user data are sent is defined to be “cleared”, that is, set to all zero Fill Style Indices. After that, the CDS is not to “clear” the buffer except on specific command (i.e. NOT in response to a “buffer complete” flag). This implies that double-buffering must implement a front-to-back copy on swap.

3.4.1.2 Fill Style Index Values

A Fill Style Index is an unsigned 8-bit value that is used to select a graphic representation (fill style) from a pre-defined table for use in filling an area on a layer. Because fill styles depend heavily on CDS hardware capabilities, and because they are “look-and-feel”, they are not further defined in this specification.

COMMENTARY

The actual fill styles used will depend on both the CDS hardware capability and the supplier/airframe manufacturer/system integrator/ customer preference for look-and-feel. A fill style may be a solid color fill, as is typical of late-1990’s weather radar displays, or it may be a patterned fill, as is typical of late-1990’s terrain displays, or it may incorporate alpha (transparency) level or other visual attributes of which modern graphics hardware is capable.

Equipment suppliers will need to agree how to map these 8-bit values to available hardware capabilities, and assign specific values to the real-world meaning. In some CDS implementations, the allowable range of values may be smaller than 0 to 255, or the indices may have sub-fields. In some CDS implementations, each UA might have its own palette. In others, all UAs might share a global fill palette.

3.4.2 ExternalSource

Categories:
None

Description:
The function of the ExternalSource widget is to specify to the CDS where an external input should appear on the display. For example, an external input may be a video signal input or a bitmap image. Note that if a UA wants to display video on the CDS, video input processing provisions are necessary in the CDS. The existence of this widget in this standard does not guarantee that it will be possible to display a video or a bitmap image. The following points must be clearly understood:

- The integrator and the CDS supplier define how an external input stream is to be sent and processed by the display
- The integrator knows the limitations of the CDS for processing of these input streams. For instance, the CDS may not **be able to** re-size or rotate **the received** video signal.

  Note: the ExternalSource widget is unique in the sense that it is necessary for the UA supplier and the integrator to define the specific method to bring video to the display.

Restriction:
None

ExternalSource Parameters are defined in Table 3.4.2-1.

Table 3.4.2-1 – ExternalSource Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EXTERNALSOURCE
WidgetIdent	D	Unique identifier of the widget.
Parent Identifier	D	Identifier of the immediate container of the widget
Specific parameters		
Visible	DR	Visibility of the widget
Xpos	D	The X position of the widget reference point
Ypos	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
SourceReference	D	Identifier of input stream source reference available on the CDS and used by the UA.
SourceX	DR	For those channels that support zoom/pan/scale/clipping/etc, this parameter indicates the origin within the source image for display
SourceY	DR	
SourceDX	DR	For those channels that support zoom/pan/scale/clipping/etc, this parameter indicates the extent within the source image for display
SourceDY	DR	
StyleSet	DR	Might control transparency, zoom/clip, etc.

ExternalSource Creation Structure is defined in Table 3.4.2-2.

Table 3.4.2-2 – ExternalSource Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EXTERNALSOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	uchar	8	
X Pos	long	32	
Y Pos	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SourceReference	ushort	16	
UnusedPad	ushort	16	
SourceX	ulong	32	
SourceY	ulong	32	
SourceDX	ulong	32	
SourceDY	ulong	32	
StyleSet	ushort	16	
UnusedPad	ushort	16	

No event is associated with the ExternalSource widget.

Available SetParameter identifiers and associated data structure are defined in Table 3.4.2-3.

Table 3.4.2-3 – ExternalSource Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to sections 4.5.3)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
SourceX	ulong	32	A661_SOURCE_X	A661_ParameterStructure_4Bytes
SourceY	ulong	32	A661_SOURCE_Y	A661_ParameterStructure_4Bytes
SourceXY	ulong x 2	32 x 2	A661_SOURCE_XY	A661_ParameterStructure_8Bytes
SourceDX	ulong	32	A661_SOURCE_DX	A661_ParameterStructure_4Bytes
SourceDY	ulong	32	A661_SOURCE_DY	A661_ParameterStructure_4Bytes
SourceDXDY	ulong x 2	32 x 2	A661_SOURCE_DXDY	A661_ParameterStructure_8Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes

3.4.3 MapVert

Categories:

- Container
- Map Management

Description:
The MapVert widget is the counterpart of the MapHorz widget for a vertical display made of a slice presentation. It is based on Cartesian coordinate system. Typically the horizontal axis will be distance in nautical miles and the vertical axis will be height in feet. The UA master of the vertical display will have to create such a widget and through it, provide the following information to the CDS:

- The location of the widget in the window
- The size of the widget
- The geographic correspondence of this size. From there, real world distance be converted in screen distance on both axes
- Position of a reference point both in screen coordinate and Geographic coordinates. From there the CDS can interpret absolute or relative coordinates for Items. For example, on the horizontal axis, the reference point is 30 mm from origin of widget, 30Nm in geographic coordinates. Knowing the distance equivalence, the CDS can position either an Item at 45Nm absolute or 15Nm relative to the reference point

COMMENTARY

In cases where the active areas of one or more interactive MapItem or MapSource widgets overlap, the sending of one or more events will be CDS dependent.

Restriction:
None

MapVert Parameters are defined in Table 3.4.3-1.

Table 3.4.3-1 – MapVert Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPVERT
WidgetIdent	D	Unique identifier of the widget
Parent Identifier	D	Identifier of the immediate container of the widget
Specific parameters		
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
PosX	D	The X position of the widget reference point (screen coordinate system)
PosY	D	The Y position of the widget reference point (screen coordinate system)
SizeX	D	Area size X
SizeY	D	Area size Y
RangeX	DR	Equivalent in Geographic coordinates of Area Size X
RangeY	DR	Equivalent in Geographic coordinates of Area Size Y
RefPosX	DR	Position X of the reference point, expressed in screen coordinates from

Parameters	Change	Description
		PosX
RefPosY	DR	Position Y of the reference point, expressed in screen coordinates from PosY
RefGeoPosX	DR	Position X of the reference point, expressed in geographic coordinates
RefGeoPosY	DR	Position Y of the reference point, expressed in geographic coordinates
MapSynchronizationNumber	**R**	**See section 3.2.8.4**

MapVert Creation Structure is defined in Table 3.4.3-2a.

Table 3.4.3-2a – MapVert Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_MAPVERT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
RangeX	fr(32768)	32	
RangeY	long	32	
RefPosX	long	32	
RefPosY	long	32	
RefGeoPosX	fr(32768)	32	
RefGeoPosY	long	32	

MapVert Event Structures: A661_EVT_ITEM_SYNCHRONIZATION **is** defined in Table 3.4.3-2b. This event is initiated by the transmission of an Item_Synchronization in a MapVert_ItemList. See the definition of the Item_Synchronization in the MapVert_ItemList for more details.

Table 3.4.3-2b – MapVert Event Structures: A661_EVT_ITEM_ SYNCHRONIZATION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ITEM_SYNCHRONIZATION
LinkedIdent	ushort	16	Identifier of the connector identifier link to the layer containing the MapVertItemList which has received the Item Synchronization. If no connector is used to connect the layer containing the MapVertItemList with the MapVert, this field is to be set to identifier of the MapVertItemList.
DataType	uchar	8	Data type coming from the item synchronization. For example: VD_MODE_RANGE …
UnusedPad	N/A	24	0
SynchronizationData 1st word		32	Data is implementation dependent.
SynchronizationData 2nd word		32	Data is implementation dependent.

Available SetParameter identifiers and associated data structure are defined in Table 3.4.3-3.

Table 3.4.3-3 – MapVert Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to sections 4.5.3)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
RangeX	fr(32768)	32	A661_RANGE_X	A661_ParameterStructure_4Byte
RangeY	long	32	A661_RANGE_Y	A661_ParameterStructure_4Byte
RangeX RangeY	fr(32768) long	32 x 2	A661_RANGE_XY	A661_ParameterStructure_8Bytes
RefPosX	long	32	A661_PRP_SCREEN_X	A661_ParameterStructure_4Byte
RefPosY	long	32	A661_PRP_SCREEN_Y	A661_ParameterStructure_4Byte
RefPosX RefPosY	long x 2	32 x 2	A661_PRP_SCREEN_XY	A661_ParameterStructure_8Bytes
RefGeoPosX	fr(32768)	32	A661_PRP_X	A661_ParameterStructure_4Byte
RefGeoPosY	long	32	A661_PRP_Y	A661_ParameterStructure_4Byte
RefGeoPosX RefGeoPosY	fr(32768) long	32 x 2	A661_PRP_XY	A661_ParameterStructure_8Bytes
MapSynchronizationNumber	**ushort**	**16**	**A661_MAP_SYNCHRONIZATION_NUMBER**	**A661_ParameterStructure_2Bytes See section 3.2.8.4**

3.4.4 MapVert_Source

Categories:

- Map management
- Container
- Interactive

Description:
The MapVert_Source is the equivalent of the MapHorz_Source for vertical displays. The MapVert_Source widget is a specialized container. It contains some MapVert_ItemList widgets to display Items expressed in a common coordinate system. The MapDataFormat (X or Y) parameters allow a UA to transmit its data either in as absolute values or relative to the Reference Point.

MapVert_Source is an interactive widget. The display area of the MapVert_Source is the same as the MapVert. The UA may need to receive the cursor position on a crew member validation with CCD on the MapVert_Source display area. The MapVert_Source EventFlag parameter provides a means to the map UA to control the CDS sending this event. The X,Y position sent by the CDS is expressed in the MapVert_Source coordinate system.

Restriction:
The MapVert_Source should be directly under a MapVert widget or a Layer widget. When directly attached to a Layer, the layer should not be attached to a window displayed alone.

MapVert_Source Parameters are defined in Table 3.4.4-1.

Table 3.4.4-1 – MapVert_Source Parameters

Parameters	Change	Description
WidgetType	D	A661_MAPVERT_SOURCE
WidgetIdent	D	Unique identifier of the widget
Parent Identifier	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
MapDataFormatX	D	Relative: X position of Items expressed relative to Reference point. Absolute: X position of Items expressed in absolute value. X_Dist: X fixed position with respect to screen reference point (no motion compensation)
MapDataFormatY	D	Relative: Y position of Items expressed relative to Reference point. Absolute: Y position of Items expressed in absolute value. Y_Alt: Y fixed position with respect to screen reference point (no motion compensation)
EventFlag	DR	Indicates if the UA wants to receive the cursor position upon click, expressed in its coordinate system.

MapVert_Source Creation Structure is defined in Table 3.4.4-2a.

Table 3.4.4-2a – MapVert_Source Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAP_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
MapDataFormatX	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE A661_MDF_X_DIST
MapDataFormatY	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE A661_MDF_Y_ALT
EventFlag	uchar	8	A661_FALSE A661_TRUE
Unused	uchar	8	

MapVert_Source Format Structure is defined in Table 3.4.4-2b.

Table 3.4.4-2b – MapVert_MapData Format Values

Parameter	Value	Origin	Units of Measure	Type (Note 1)	LSB
MapDataFormatX	A661_MDF_ABSOLUTE	Absolute Reference: Position with respect to RefGeoPosX is: DataPosX-RefGeoPosX	nM	long	fr(32768)
MapDataFormatX	A661_MDF_RELATIVE	Relative Reference: RefGeoPosX The Position with respect to RefGeoPosX is: DataPosX	nM	long	fr(32768)
MapDataFormatX	A661_MDF_X_DIST	RefPosX	nM	float	N/A
MapDataFormatY	A661_MDF_ABSOLUTE	Absolute Reference: Position with respect to RefGeoPosY is: DataPosY-RefGeoPosY	feet	long	1
MapDataFormatY	A661_MDF_RELATIVE	Relative Reference: RefGeoPosY The Position with respect to	feet	long	1

		RefGeoPosY is: DataPosY			
MapDataFormatY	A661_MDF_Y_ALT	RefPosY	feet	float	N/A

Note: Type refers to the data type of the X and Y parameters in the children of the MapVert_Source.

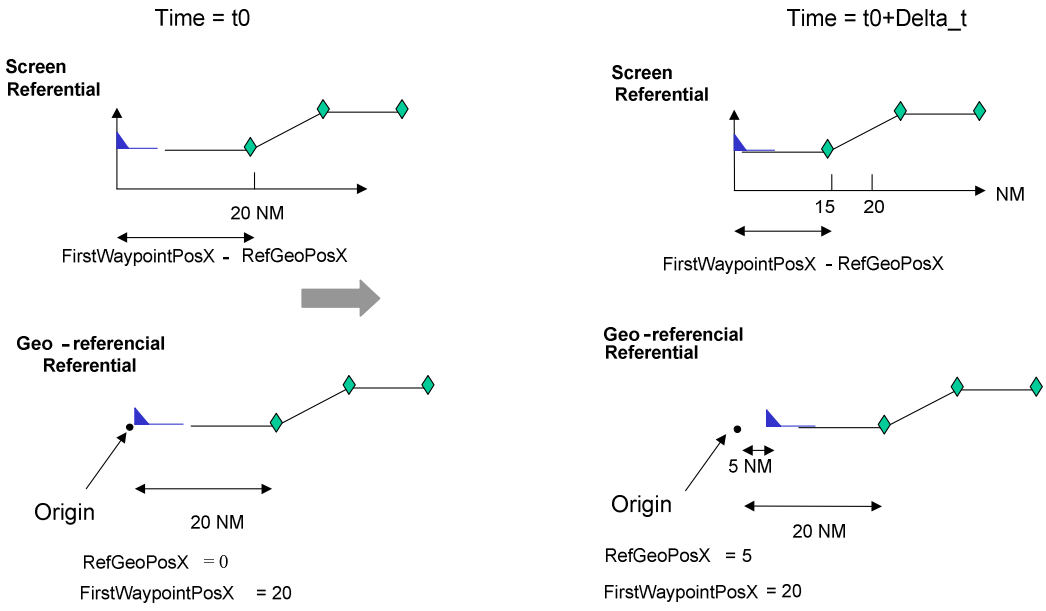


Figure 3.4.4-1 – Illustration of MapDataFormatX = A661_MDF_ABSOLUTE, X Origin is an Absolute Point

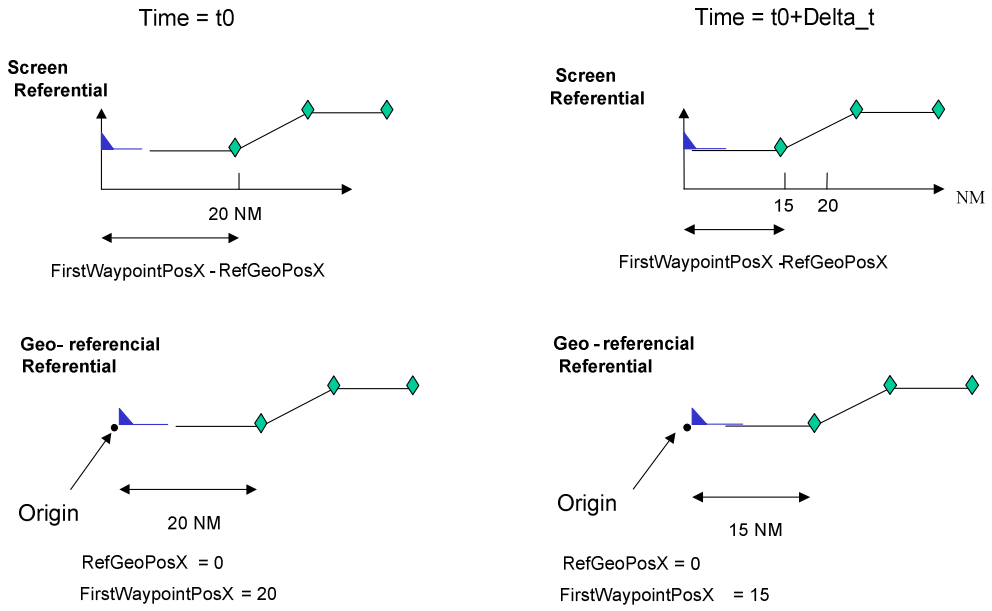


Figure 3.4.4-2 – Illustration of MapDataFormatX = A661_MDF_RELATIVE : X Origin is RefGeoPosX

Note: With MapDataFormatX = A661_MDF_RELATIVE, RefGeoPosX parameter is without effect.

MapVert_Source Event Structures: A661_EVT_SELECTION_MAP is defined in Table 3.4.4-3.

Table 3.4.4-3 – MapVert_Source Event Structures: A661_EVT_SELECTION_MAP

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION_MAP
UnusedPad	N/A	16	0
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

Available SetParameter identifiers and associated data structure are defined in Table 3.4.4-4.

Table 3.4.4-4 – MapVert_Source Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
EventFlag	uchar	8	A661_EVENT_FLAG	A661_ParameterStructure_1Byte

3.4.5 MapVert_ItemList

Categories:

- Map management
- Graphical Representation
- Interactive

Text string Description:
The MapVert_ItemList is equivalent to the MapHorz_ItemList for vertical displays. A MapVert_ItemList contains a list of Items to be drawn. This list is of fixed size specified through the maximum number of Items. The type of each Item inside the MapVert_ItemList can be modified at run-time, which makes the list dynamic. A set of parameters is associated with each type of Item (refer to Section 3.3.22.2.1, Item Structure).

One or several items can be modified through a SetParameter command with BufferOfItems as Parameter_Ident. An Item should be modified in its entirety. For instance, the X coordinate of a symbol can not be changed by itself.

Insert and delete operations are not allowed on the list. However, one specific type of Item is NOT_USED. The Item with the NOT_USED type will be ignored, i.e., is they will have no effect on the processing of following items.

Section 3.4.5.1 describes the standardized items and their functionality. Section 3.4.5.2 describes the A661_ParameterStructure to address the Items.

Restriction:
A MapVert_ItemList must be in a MapVert_Source container.

MapVert_ItemList Parameters is defined in Table 3.4.5-1.

Table 3.4.5-1 – MapVert_ItemList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPVERT_ITEMLIST
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
Specific parameters		
MaxNumberOfItem	D	Maximum number of items that the UA can address under the MapVert_ItemList.
BufferOfItems	R	Buffer of the Map Items
MapSynchronizationNumber	**R**	**See section 3.2.8.4**
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

MapVert_ItemList Creation Structure is defined in Table 3.4.5-2a.

Table 3.4.5-2a – MapVert_ItemList Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPVERT_ITEMLIST
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfItem	ushort	16	
UnusedPad	N/A	16	0

MapVert_ItemList Event Structures: A661_EVT_SELECTION is defined in Table 3.4.5-2b.

Table 3.4.5-2b – MapVert_ItemList Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SELECTION
Item Index	ushort	16	Index of the item that has been selected. Index from 1 to MaxNumberOfItem.

Available SetParameter identifiers and associated data structure are defined in Table 3.4.5-3.

Note: The structure of this event is different than other A661_EVT_SELECTION events in that it does not contain a pad.

Table 3.4.5-3 – MapVert_ItemList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
BufferOfItems	N/A	{32}	A661_BUFFER_OF_MAPVERT_ITEMS	A661_ParameterStructure_BufferOfItems Refer to "MapVert_ItemList A661_ParameterStructure Specifics" Section 3.4.5.2, **especially Section 3.4.5.2.2.**
MapSynchronizationNumber	**ushort**	**16**	**A661_MAP_SYNCHRONIZATION_NUMBER**	**A661_ParameterStructure_2Bytes See section 3.2.8.4**
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.4.5.1 MapVert_ItemList Standard Items Description

This section describes all the Item structures.

Table 3.4.5.1 – MapVert_ItemList Standard Items Description

Name of Item	Function
FILLED_POLY_START	This Item is used to signify the start of a closed, filled polygon definition. It holds X/Y parameters, like LINE_START, and a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items (up to the EndFlag) define the vertices and edges of a polygon that is closed and filled with the indicated fill style.
ITEM_STYLE	For drawing any symbol or line, the CDS must apply the last defined ITEM_STYLE in the list. If no ITEM_STYLE has been defined, the CDS will apply the default ITEM_STYLE.
ITEM_SYNCHRONIZATION	This item has been defined to attach **frame data** to symbology expressing **the** context of the computation or the rendering for the symbology frame. This data is sent in A661 in order to avoid synchronization issue**s** between the symbology frame and the attached information (e.g. mode, range; MRP …). This item can be used to pass synchronization information from the application owning a MapVert Item List and the application owning the parent MapVert.
LEGEND	This Item is used to store Legend Strings. Some symbols may contain logic to automatically position legends. LEGEND Items will then follow the SYMBOL Item and carry this legend. Each LEGEND Item can only hold 16 characters including the NULL character. Several LEGENDS Item can be used to carry longer strings. CR is recognized as either NextField (For symbols with automatic Legend positioning) or as a normal Carriage Return / Line Feed if LEGEND follows a LEGEND_ANCHOR. The last LEGEND Item of a group must have its EndFlag set.

Name of Item	Function
LEGEND_ANCHOR	This Item is used to specify the position of a LEGEND not attached to a symbol.
LEGEND_POP_UP	This Item is a basic LEGEND, but it will appear only when the crew member selects the associated SYMBOL_x Item. Disappearance of the LEGEND_POP_UP is airframe manufacturer/system integrator specification dependent.
LINE_START	This Item is used to signify the start of a line. It holds only X/Y parameters, interpreted by the CDS depending on the MapVert_Source DataFormat
LINE_SEGMENT	This Item is used to draw a line, using the last defined style in the list, from the previous LINE_xxx End position, to the specified X/Y coordinates. This Item holds EndFlag, set if it is the last item of a line.
NOT_USED	This Item is used when the Item is to be discarded by the CDS. There is no effect on subsequent Items interpretation.
SYMBOL_GENERIC	This Item represents the basic symbol, which holds X/Y parameters along with a type of symbol and possibly an EndFlag. Some of these types may include an Automatic Legend positioning. In this case, and provided the EndFlag is not set on the symbol, the CDS will interpret the following LEGEND Items as part of the symbol legend. When multiple Fields exist on the symbol, “Carriage Return” will signify to the CDS that a field end is reached.
SYMBOL_ROTATED	Same than SYMBOL_GENERIC except an orientation parameter is added.
SYMBOL_RUNWAY	Same than SYMBOL_GENERIC except Length parameter is added
TRIANGLE_STRIP _START	**This Item is used to signify the start of a closed, filled polygon defined by a series of triangle strips.**
TRIANGLE_FAN _START	**This Item is used to signify the start of a closed, filled polygon defined by a series of triangles arranged in a fan.**
TRIANGLE_SEGMENT	**Defines a single vertex of a Triangle Strip or Triangle Fan.**
TRIANGLE_SEGMENT_ DOUBLE	**Defines two vertices of a Triangle Strip or Triangle Fan.**
TRIANGLE_END	**Defines the last vertex of a Triangle Strip or Triangle Fan.**
TRIANGLE_END _DOUBLE	**Defines the last two vertices of a Triangle Strip or Triangle Fan.**

3.4.5.2 MapVert_ItemList A661_ParameterStructure Specifics

This section describes the A661_ParameterStructure_BufferOfItems **for MapVert_ItemList**.

3.4.5.2.1 Item Structures

All the structures include the same format, three fields for the first 4-byte word. One field is not used on all Items, however it is maintained for consistency.

3.4.5.2.1.1 Item_Style

Item_Style is defined in Table 3.4.5.2.1.1.

Table 3.4.5.2.1.1 – Item_Style

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_STYLE
UnusedPad	N/A	8	0
ItemStyleSet	ushort	16	
UnusedPad	N/A	16	0

3.4.5.2.1.2 Legend_Anchor

Legend_Anchor is defined in Table 3.4.5.2.1.2.

Table 3.4.5.2.1.2 – Legend_Anchor

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

3.4.5.2.1.3 Legend and Legend_Pop_Up

This Item must follow XXX_SYMBOL, LEGEND_ANCHOR or another LEGEND Item. The LegendString can contain special characters, line feed and carriage return. The type of symbol attached to this legend defines the position and the format of this String under control of the CDS. If LEGEND is followed by another LEGEND, they should be considered as one unique Legend, possibly including carriage return and line feed characters. The full entire LegendString (possibly across multiple Legend MapVert_Items) must have a NULL terminator.

Legend and Legend_Pop_Up is defined in Table 3.4.5.2.1.3.

Table 3.4.5.2.1.3 – Legend and Legend_Pop_Up

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND A661_LEGEND_POP_UP
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendString	{uchar}+ (not 'string')	{32}+	Max 16 characters including NULL and pad Followed by zero, one, two or three extra NULL for alignment of 32 bits. It must have a NULL terminator.

3.4.5.2.1.4 Line_Start

Line_Start is defined in Table 3.4.5.2.1.4.

Table 3.4.5.2.1.4 – Line_Start

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_START
UnusedPad	N/A	8	0
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

3.4.5.2.1.5 Line_Segment

Line_Segment is defined in Table 3.4.5.2.1.5.

Table 3.4.5.2.1.5 – Line_Segment

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_SEGMENT
EndFlag	uchar	8	A661_TRUE A661_FALSE
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

3.4.5.2.1.6 Not_Used

Not_Used is defined in Table 3.4.5.2.1.6.

Table 3.4.5.2.1.6 – Not_Used

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_NOT_USED
UnusedPad	N/A	8	0

3.4.5.2.1.7 Symbol_Generic

Symbol_Generic is defined in Table 3.4.5.2.1.7.

Table 3.4.5.2.1.7 – Symbol_Generic

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_GENERIC
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	uchar	16	SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
RelativePosition	N/A	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

3.4.5.2.1.8 Symbol_Runway

Symbol_Runway is defined in Table 3.4.5.2.1.8.

Table 3.4.5.2.1.8 – Symbol_Runway

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RUNWAY
EndFlag	uchar	8	A661_TRUE A661_FALSE
X		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Length	fr(32768)	32	Length of runway (in feet)

3.4.5.2.1.9 Filled_Poly_Start

There are restrictions on the polygons to be filled. The number of line segments is limited to three segments (triangle) or four segments (quadrilateral). The vertices are specified in counter-clockwise order. The polygon must be convex.

If any error is found in the polygon definition, the CDS should send an A661_ERR_SET_ABORTED exception event. The airframe manufacturer/system integrator free data field may include the ItemIndex, etc., to identify the error further.

Filled_Poly_Start is defined in Table 3.4.5.2.1.9.

Table 3.4.5.2.1.9 – Filled_Poly_Start

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_FILLED_POLY_START
FillStyleIndex	uchar	8	
X / Lat / Range		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.
Y / Lng / Angle / Alt		32	Type and LSB are expressed in map source coordinate system. Reference table 3.4.4-2b.

3.4.5.2.1.10 Item_Synchronization

Table 3.4.5.2.1.10 – Item_Synchronization

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_SYNCHRONIZATION
DataType	uchar	8	For example: VD_MODE_RANGE MRP latitude / longitude…
SynchronizationData 1st word		32	Data is implementation dependent.
SynchronizationData 2nd word		32	Data is implementation dependent.

3.4.5.2.1.11 Symbol_Rotated

Table 3.3.22.2.1.11 – Symbol_Rotated

Name	Type	Size (bits)	Value/Range When Necessary
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_ROTATED
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	For example, SYMBOL_AIRCRAFT
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X	Scaled Integer	32	First coordinate of symbol, (fixed real LSB depends on MapVert_Source MapDataFormat and RelativePosition)
Y	Scaled Integer	32	Second coordinate of symbol, (fixed real LSB depends on MapVert_Source and RelativePosition)
Orientation	fr(180)	32	Orientation of Symbol (counter-clockwise is positive orientation). Orientation is not adjusted by the CDS to account for the scaling of the RangeX and RangeY in the MapVert widget.

3.4.5.2.1.12 Triangle Strip Start

The Triangle Strip Start, Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double MapVert_ItemList Items are meant to define triangle strips, similar to those defined in SDL Symbols.

Each Triangle Strip is made up of one Triangle Strip Start, any number of Triangle Segment and Triangle Segment Double items, and one Triangle End or Triangle End Double. The Triangle Segment Double and Triangle End Double items exist to minimize MapVert_ItemList size and should be used whenever possible. See the illustration below for more details.

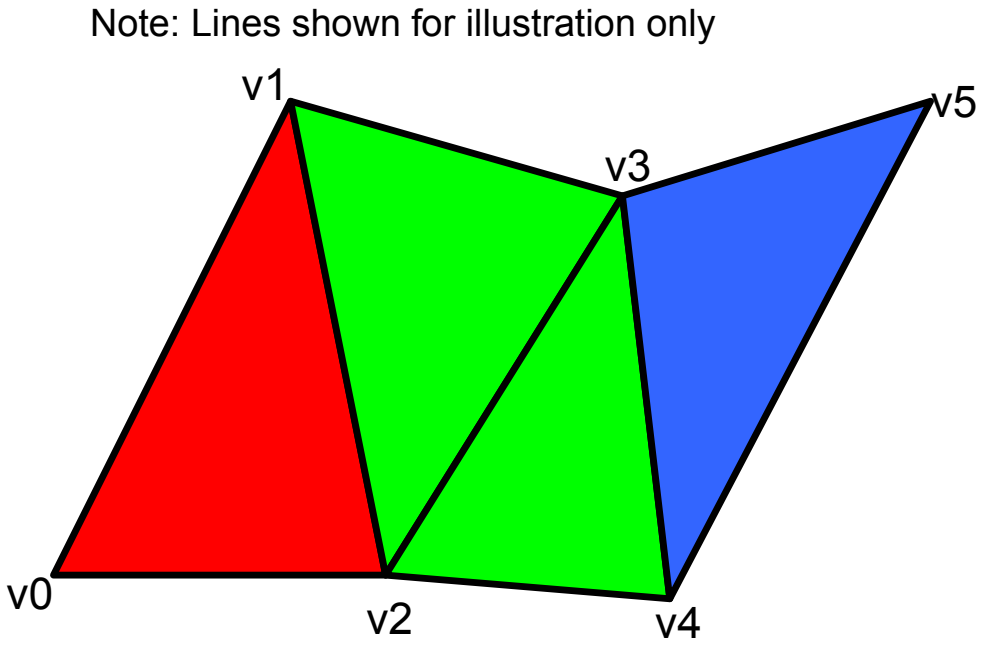


Figure 3.4.5.2.1.12 – Triangle Strip

Table 3.4.5.2.1.12 – Triangle Strip Start

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_STRIP_START
Pad	NA	8	Unused Pad
X/Lat/Rng 1	long	32	First Coordinate of first vertex
Y/Lng/Brg 1	long	32	Second Coordinate of first vertex
X/Lat/Rng 2	long	32	First Coordinate of second vertex
Y/Lng/Brg 2	long	32	Second Coordinate of second vertex

3.4.5.2.1.13 Triangle Segment

The color of the triangle completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.4.5.2.1.13 – Triangle Segment

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_SEGMENT
FillStyleIndex	uchar	8	Color for the triangle completed by this point
X/Lat/Rng	long	32	First coordinate of vertex
Y/Lng/Brg	long	32	Second coordinate of vertex

3.4.5.2.1.14 Triangle Segment Double

The color of the triangles completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.4.5.2.1.14 – Triangle Segment Double

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_SEGMENT_ DOUBLE
FillStyleIndex	uchar	8	Color for the triangles completed by these points
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

3.4.5.2.1.15 Triangle End

The color of the triangle completed by this item shall be defined by the FillStyleIndex parameter.

Table 3.4.5.2.1.15 – Triangle End

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_END
FillStyleIndex	uchar	8	Color for the triangle completed by this point
X/Lat/Rng	long	32	First coordinate of vertex
Y/Lng/Brg	long	32	Second coordinate of vertex

3.4.5.2.1.16 Triangle End Double

The color of the triangles completed by this item shall be defined by the FillStyleIndex parameter.

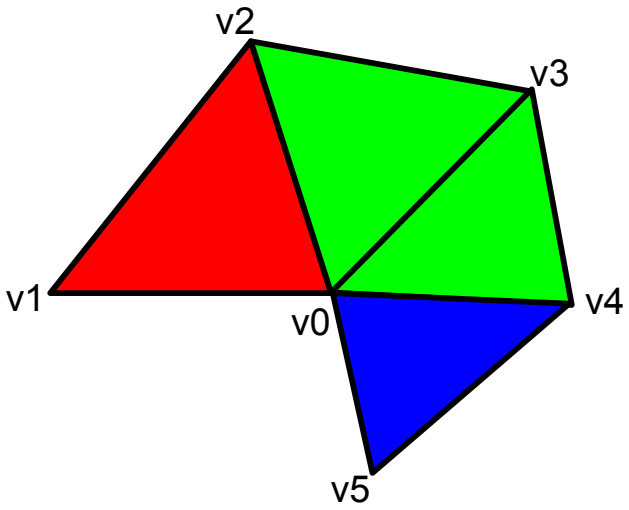
Table 3.4.5.2.1.16 – Triangle End Double

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_END_DOUBLE
FillStyleIndex	uchar	8	Color for the triangles completed by these points
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

3.4.5.2.1.17 Triangle Fan Start

The Triangle Fan Start (along with Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double) MapVert_ItemList item is meant to define triangle fans, similar to those defined in SDL Symbols. Each Triangle Fan is made up of one Triangle Fan Start, any number of Triangle Segment and Triangle Segment Double items, and one Triangle End or Triangle End Double. The Triangle Segment Double and Triangle End Double items exist to minimize MapVert_ItemList size and should be used whenever possible.

Note: Black lines shown for illustration only



Red Triangle:
Triangle Fan Start(v0, v1)
Triangle Segment(v2)
Green Triangles:
Triangle Segment Double(v3, v4)
Blue Triangle:
Triangle End(v5)

Figure 3.4.5.2.1.17 – Triangle Fan

Note: If a triangle is defined as three co-linear points, nothing will be drawn. However, the next triangle will be defined using the first and third points of the previous triangle, as usual.

Table 3.4.5.2.1.17 – Triangle Fan Start

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	
Item Type	uchar	8	A661_TRIANGLE_FAN_START
Pad	NA	8	Unused Pad
X/Lat/Rng 1	long	32	First coordinate of first vertex
Y/Lng/Brg 1	long	32	Second coordinate of first vertex
X/Lat/Rng 2	long	32	First coordinate of second vertex
Y/Lng/Brg 2	long	32	Second coordinate of second vertex

Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double are already defined (this was done in conjunction with Triangle Strip Start).

Note: The CDS will process subsequent Triangle Segment, Triangle Segment Double, Triangle End, and Triangle End Double items appropriately based on the type of triangle start item that preceded them. That is, if a Triangle Fan Start item begins a sequence of triangle items, the first and third vertex of the previous triangle is used as the base for the triangle that follows. If a Triangle Strip Start item begins a sequence of triangle items, the second and third vertex of the previous triangle is used as the base for the triangle that follows.

3.4.5.2.2 A661_ParameterStructure_BufferOfItems

A66l_ParameterStructure_BufferOfItems **as used for MapVert_ItemList** is defined in Table 3.4.5.2.2.

Table 3.4.5.2.2 – A661_ParameterStructure_BufferOfItems

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_BUFFER_OF_MAPVERT_ITEMS
ClearFlag	1	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
Number of Items	15	Number of Items modified by the command
{ItemStructures}+	{32}+	

3.4.5.3 MapVert_ItemList Interactive Items

MapVert interactive items operate the same as Map_Horz interactive items. See section 3.3.22.3 for more information on interactive map items.

3.4.6 EditBoxMultiLine

Categories:

- Graphical representation
- Interactive
- Text **String**

Description:
EditBoxMultiLine is a text edit box for displaying text across several lines in a scrolling area. The text string can be modified by the crew. When the EditBoxMultiLine is in edit mode, the CDS only reports the confirmed text string (after a crew member validation). The purpose of this widget is to allow free text edit and perform automatic line feed and scroll management.

Restriction:

None

EditBoxMultiLine **Parameters** are defined in Table 3.4.6-1.

Table 3.4.6-1 – EditBoxMultiLine Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_MULTILINE
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**
MaxStringLength	D	Maximum length of the entire text included the first NULL character ended the string (in bytes), MaxStringLength > 1.

Parameters	Change	Description
LabelString	DR	Text of the edit box
Alignment	D	Justification of the label text within the edit box area CENTER LEFT RIGHT
VerticalScroll	D	Position of scroll controls, absent/left/right/bottom/top
StartCursorPos	DR	Start position of the cursor in field when entering in edit

EditBoxMultiLine Creation Structure is defined in Table 3.4.6-2.

Table 3.4.6-2 – EditBoxMultiLine Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_MULTILINE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
Vertical Scroll	uchar	8	A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT A661_ABSENT
LabelString	string	{8}+	Followed by zero, one, two or three extra NULL for alignment on 32 bits.

The specific event sent by the EditBoxMultiLine to the owner application is defined in Tables 3.4.6-3, 3.4.6-4, 3.4.6-5 and 3.4.6-6.

Table 3.4.6-3 – EditBoxMultiLine Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

Table 3.4.6-4 – EditBoxMultiLine Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

Table 3.4.6-5 – EditBoxMultiLine Event Structure: A661_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_STRING_CONFIRMED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

Table 3.4.6-6 – EditBoxMultiLine Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

EditBoxMultiLine Runtime Modifiable Parameters is defined in Table 3.4.6-7.

Table 3.4.6-7 – EditBoxMultiLine Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StartCursorPos	ushort	16	A661_CURSOR_POS	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
LabelString	string	{32}+	A661_STRING	A661_ParameterStructure_String
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte

3.4.7 ComboBoxEdit

Categories:

- Graphical representation
- Interactive
- Text **String**

Description:
Like ComboBox, ComboBoxEdit provides a means to select one item in a list of items. This widget is composed of a static part displaying the selected item and a pop up part displaying possible items. The number of the current selected entry is held in the SelectedEntry parameter. The complete list of possible Entries is held in a string array (parameter EntryList). The list is displayed upon crew member selection (e.g., a click on the arrow button associated **with** the Selected Entry).

Moreover, ComboBoxEdit allows the crew to enter **new data to** the static part **of the widget**. When ComboBoxEdit is in edit mode, the CDS may report all modification done on the edited string and the final confirmed string, or only report the confirmed string (after a crew member validation). This option may be set by the UA through the “ReportAllChanges” parameter. If ReportAllChanges is True and, after having entered a text, the crewmember finally aborts the edit, the CDS should send a specific event to the UA with the former validated LabelString as parameter of the event.

Note that SelectingAreaHeight and the SelectingAreaWidth represent the Y and X Size of the pop-up part of the ComboBoxEdit

OpeningMode of the ComboBoxEdit is used to determine how the ComboBox opens.

The pop-up part of the ComboBoxEdit is displayed on top of its containing window and is affected by the clipping area of its containing window.

Restriction:
N/A

ComboBoxEdit Parameters are defined in Table 3.4.7-1.

Table 3.4.7-1 – ComboBoxEdit Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_COMBO_BOX_EDIT
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget (in the closed mode)
SizeY	D	The Y dimension size (height) of the ComboBoxEdit (in the closed mode)

Parameters	Change	Description
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaWidth	D	X Size of the area available to display the entry list (when the ComboBoxEdit is opened)
SelectingAreaHeight	D	Y Size of the area available to display the entry list (when the ComboBoxEdit is opened)
OpeningMode	D	Mode of combo opening: UP CENTERED DOWN
MaxStringLength	D	Maximum string length for each entry item (including end tag character) but also for any user caption process, MaxStringLength > 1.
Alignment	D	Justification of the label text within the edit area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**
StartCursorPos	DR	Start position of the cursor in field when entering in edit
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be lower than MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list from 1 to NumberOfEntries if an entry is selected and 0 else
OpeningEntry	DR	Entry number which is ensured to be visible when the ComboBoxEdit is opened. Opening entry is in the range [0; NumberOfEntries]

Parameters	Change	Description
		OpeningEntry will be set to 0, if not used.
LabelString	N/A	Text of the new entry entered by the crewmember
EntryList [MaxEntryNumber]	DR	String array holding the list of entries.

Note: D is Design time. R is Run time.

N/A means that this parameter is only used as an event value. It is never set by the UA (not at definition time nor at runtime).

ComboBoxEdit Creation Structure is defined in Table 3.4.7.-2.

Table 3.4.7-2 – ComboBoxEdit Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_COMBO_BOX_EDIT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
OpeningEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN
AutomaticFocusMotion	uchar	8	
StartCursorPos	ushort	16	
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
EntryList [NumberOfEntries]	string[]	8 * string length + PAD	Each string is ended by character NULL (used as string separator). The complete string list is followed by zero, one, two or three NULL character(s) to be 32 bits aligned

The specific events sent by the ComboBoxEdit to the owner application are:

Table 3.4.7-3 – ComboBoxEdit Event Structures: A661_EVT_SELECTED_ENTRY_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SEL_ENTRY_CHANGE
SelectedEntry	ushort	16	

Table 3.4.7-4 – ComboBoxEdit Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

Table 3.4.7-5 – ComboBoxEdit Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

Table 3.4.7-6 – ComboBoxEdit Event Structures: A661_EVT_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CONFIRMED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

Table 3.4.7-7 – ComboBoxEdit Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.4.7-8 – ComboBoxEdit Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	16	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
StartCursorPos	ushort	16	A661_CURSOR_POS	A661_ParameterStructure_2Bytes
EntryList [NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray
OpeningEntry	ushort	16	A661_OPENING_ENTRY	A661_ParameterStructure_2Bytes
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte

3.4.8 MenuBar

Categories:
Container

Description:
A MenuBar is a widget containing PushButtons, PicturePushButtons and PopUpMenuButtons. It implements specific behaviors to move from one button to another.

COMMENTARY

Example behavior of MenuBar:

When a PopUpMenu attached to a button is visible, a validation through the cursor on another button of the MenuBar closes the PopUpMenu and activates the selected button. Right / Left arrow keys move the focus from one button to another button of the MenuBar.

The Buttons contained in the MenuBar are individually defined with the MenuBar as parent widget. The positions of buttons inside the MenuBar are defined by their own parameters PosX and PosY according to the following rules:

- For Horizontal menu bar, all buttons inside the MenuBar have the same PosY and SizeY defined by the menu bar parameter ButtonPos and ButtonSize
- For Vertical menu bar, all buttons inside the MenuBar have the same PosX and SizeX defined by the menu bar parameter ButtonPos and ButtonSize

- Restriction:
  A MenuBar has only children types:

- PushButton
- PicturePushButton
- PopupMenuButton

MenuBar parameters are defined in Table 3.4.8-1.

Table 3.4.8-1 – MenuBar Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MENU_BAR
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
Specific parameters		
Horizontal	D	True: MenuBar is horizontal False: MenuBar is Vertical
ButtonPos	D	If Horizontal =True: Value of the buttons parameter PosY If Horizontal =False: Value of the buttons parameter PosX
ButtonSize	D	If Horizontal =True: Value of the buttons parameter SizeY If Horizontal =False: Value of the buttons parameter SizeX

MenuBar Creation Structure is defined in Table 3.4.8-2.

Table 3.4.8-2 – MenuBar Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_MENU_BAR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
Horizontal	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	
PosX	long	32	
PosY	long	32	
ButtonPos	long	32	
ButtonSize	long	32	

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.4.8-3 – MenuBar Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte

3.5 Widget Extension (Supplement 2)

This section was added in Supplement 2. It introduces new widgets to ARINC 661.

3.5.1 MutuallyExclusiveContainer

Categories:
Container

Description:
The MutuallyExclusiveContainer has no graphical representation. Its purpose is to group children widgets and to provide a means for managing the exclusive display of the children. The operation of the MutuallyExclusiveContainer is very similar to that of the BasicContainer except that only one child of the MutuallyExclusiveContainer can be visible. All of the children of a BasicContainer can be visible.

A UA can make a child visible in the container by setting the VisibleChild parameter. Only the Widget in the container that has the WidgetIdent that matches the VisibleChild will be processed for display. The visibility of the child is controlled by the child's Visible parameter and is not changed by changing the VisibleChild parameter. For a child in the container to be visible the VisibleChild must be set to the child's WidgetIdent and the Visible parameter for the Widget must be True. Normal rules for visibility and enabling of a parent widget apply to the MutuallyExclusiveContainer.

The contained widgets are positioned with respect to the PosX and PosY of the MutuallyExclusiveContainer. The MutuallyExclusiveContainer does not have clipping capabilities.

There is no interactivity associated with this widget so there are no events associated with this widget. Only the UA controlling the layer can change the VisibleChild.

The following figure shows the difference between the MutuallyExclusiveContainer and the BasicContainer. The bottom picture shows an example application of the MutuallyExclusiveContainer to make one message visible from a collection of 4 messages.

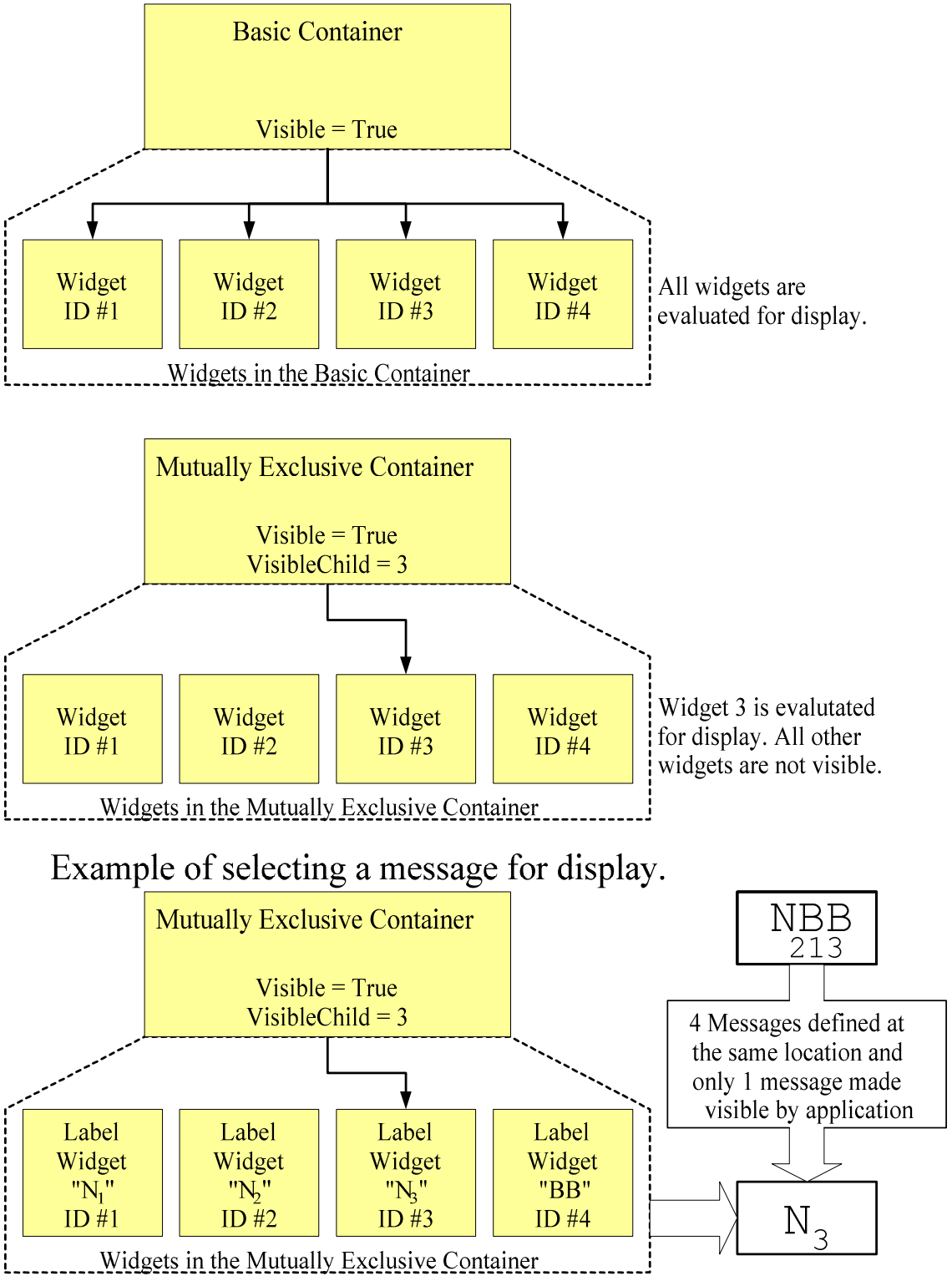


Figure 3.5.1-1 – Mutually Exclusive Container

Restriction:

None

MutuallyExclusiveContainer Parameters are defined in Table 3.5.1-1.

Table 3.5.1-1 – MutuallyExclusiveContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MUTUALLY_EXCLUSIVE_CONTAINER
WidgetIdent	D	Unique identifier of the widget
Parent Identifier	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
Specific parameters		
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
VisibleChild	DR	Identifier of the widget to be made visible. A value of 0 means that no widget is visible

MutuallyExclusiveContainer Creation Structure is defined in Table 3.5.1-2.

Table 3.5.1-2 – MutuallyExclusiveContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_Mutually_Exclusive_Container
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
VisibleChild	ushort	16	
UnusedPad	N/A	16	

Available SetParameter identifiers and associated data structure are:

Mutually Exclusive Container Runtime Modifiable Parameters are defined in Table 3.5.1-3.

Table 3.5.1-3 – MutuallyExclusiveContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
VisibleChild	ushort	16	A661_VISIBLE_CHILD	A661_ParameterStructure_2Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes

3.5.2 ProxyButton

Categories:
Interactive

Description:
The ProxyButton directs a physical button press as a select event to the target widget. If the target widget is visible and enabled at the time of the physical button press, it will respond in the same way as if the user had selected the widget. The CDS will supply a list of unique identifiers for the selection keys available in the system. A common use for this feature is a bezel line select soft key. If the target widget ID is 0 then the select event is sent directly to the UA.

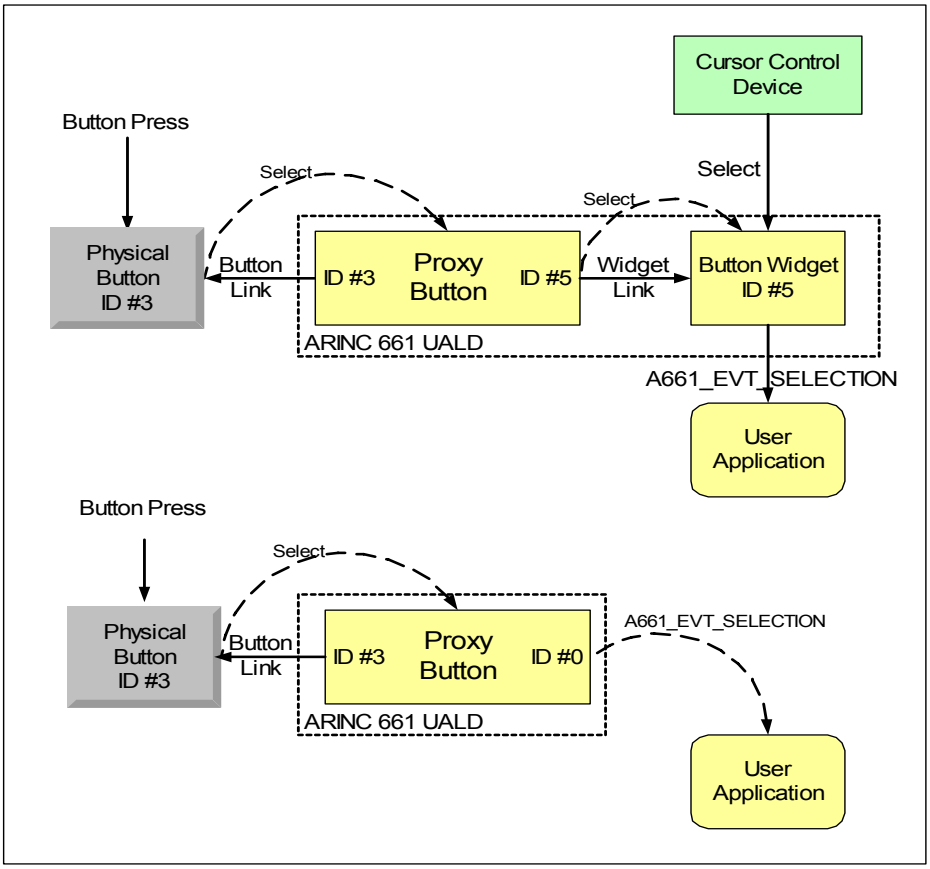


Figure 3.5.2-1 – Proxy Button

Restrictions:

- The result of assigning more than one ProxyButton widget to a single key will cause events to be sent to multiple widgets when that key is pressed
- If an undefined target widget ID is set the A661_ERR_SET_ABORTED may be sent to the application
- If the ProxyButton is in a container or layer that is disabled or not visible then the ProxyButton is disabled. When the ProxyButton is disabled, the ProxyButton will not pass events from physical buttons

ProxyButton Parameters are defined in Table 3.5.2-1.

Table 3.5.2-1 – ProxyButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PROXY_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be activated
Specific Parameters		
DedicatedKeyIdent	D	The unique identifier of the dedicated CDS key
TargetWidgetIdent	DR	Identifier of the widget to receive the select event

ProxyButton Creation Structure is defined in Table 3.5.2-2.

Table 3.5.2-2 – ProxyButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_PROXY_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
UnusedPad	N/A	8	0
DedicatedKeyIdent	ushort	16	
TargetWidgetIdent	ushort	16	

ProxyButton Event Structures:

This event indicates to the UA that a crew member has interacted with the widget.

ProxyButton Event Structures: A661_EVT_SELECTION is defined in Table 3.5.2-3.

Table 3.5.2-3 – ProxyButton Event Structures: A661_EVT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	**ushort**	16	A661_EVT_SELECTION
UnusedPad	**N/A**	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.5.2-4.

Table 3.5.2-4 – ProxyButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParamterStructure	Type of Structure Used (Refer to 4.5.4.5)
TargetWidget Ident	ushort	16	A661_TARGET_WIDGET_ID	A661_Parameterstructure_2bytes
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte

3.5.3 WatchdogContainer

Categories:
Container

Description
The WatchdogContainer widget is a non-graphical container widget stimulated, during normal operation, by the UA at a periodic rate. One or more widgets must be placed into the container and the ShowIfFailIdent must be set to either zero or the value of one of the widgets in the container. During normal operation, all of the widgets within this container, except for the widget referenced with the ShowIfFailIdent, are evaluated for display. If the UA fails to stimulate the Refresh parameter in the Watchdog Container widget for FailCountLimit periods then the watchdog is considered expired causing an event to be sent back to the UA and causing the CDS to display the widget referenced by ShowIfFailIdent.

This widget is useful when used in combination with the BufferFormat widget to assure that a data set is updated at a specific rate. The BufferFormat contains a set of functionally related parameters and the “Refresh” parameter for a Watchdog widget. If the BufferFormat is updated at the period defined by “TimePeriod” then the timer is satisfied and no action is taken. If the “Refresh” is not set to true for FailCountLimit periods, the timer expires and two actions are taken by the CDS:

- An event is sent back to the UA indicating that the timer has expired
- The Widget referenced by ShowIfFailIdent is evaluated for display. If the Visible parameter of the child widget is True then the widget is displayed

If the UA starts setting the “Refresh” to true for “ValidCountLimit” periods, the Watchdog is satisfied. When the Watchdog is satisfied the following actions are taken by the CDS:

- An event is sent back to the UA indicating the watchdog is satisfied
- The widget referenced by ShowIfRefreshIdent is evaluated for display. If the Visible parameter of the child widget is True then the widget is displayed

The timer of a watchdog that is a child of another watchdog is still updated regardless of the state of the parent watchdog.

In the example below, the first picture shows a buffer format updating the watchdog timer to display widget #12. The Refresh parameter is included in the buffer format. The bottom picture shows the case when the buffer format is no longer received by the CDS. In this case the fail indication in Widget #24 is displayed.

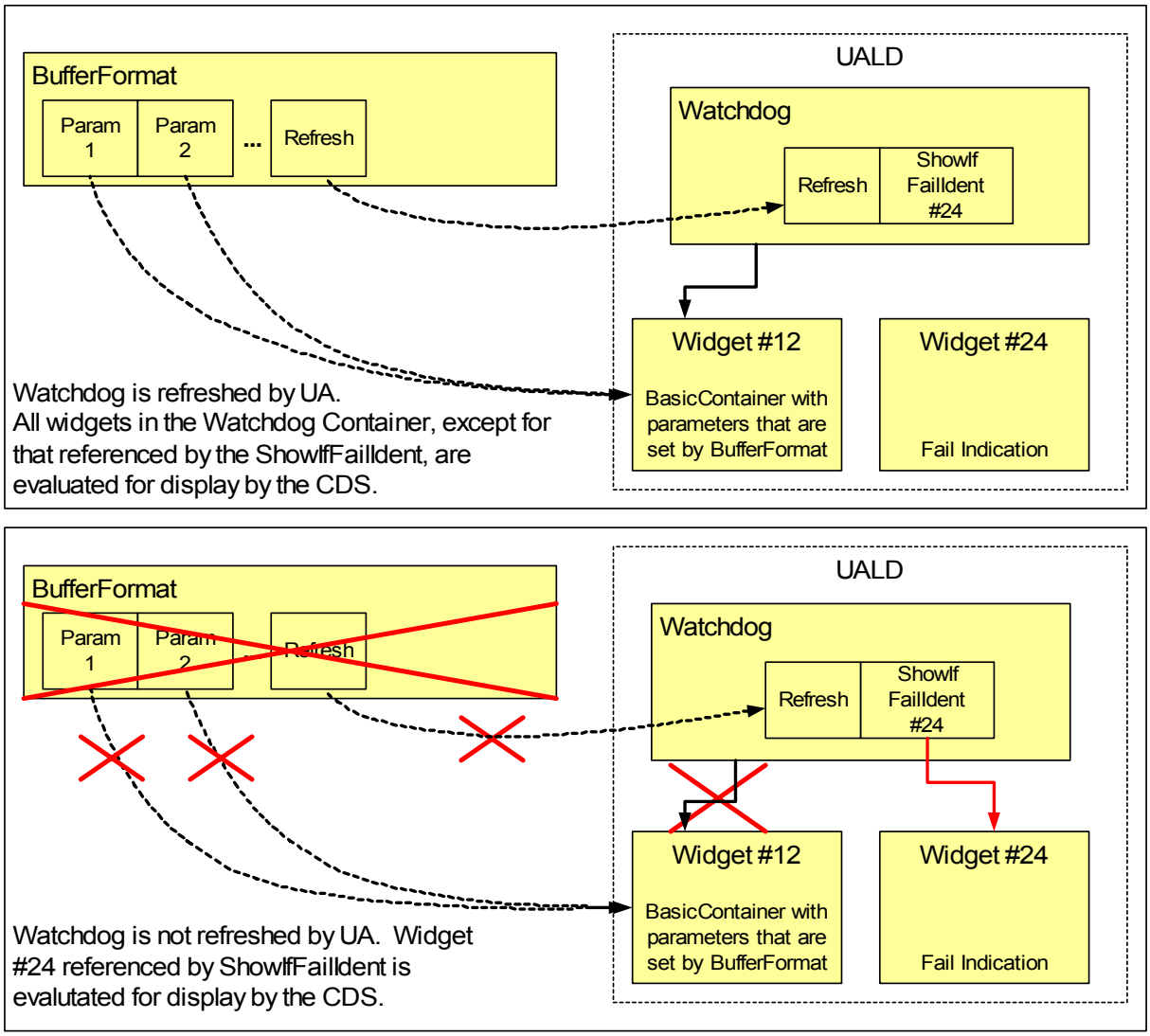


Figure 3.5.3-1 – Watchdog Buffer

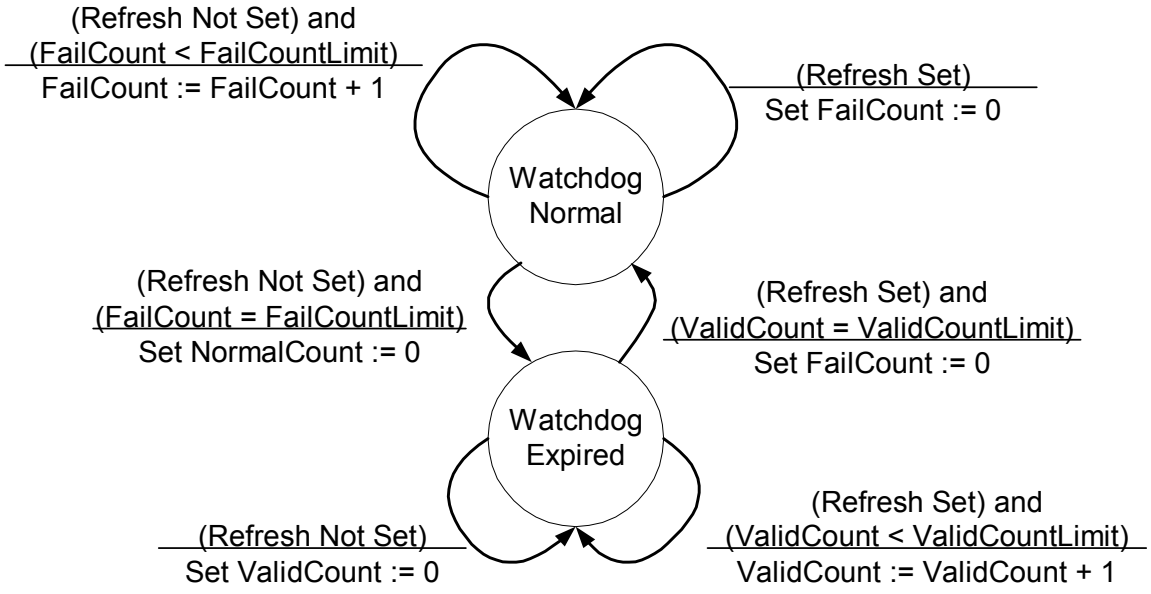
To provide a better understanding of how the timer works consider the following example:

- TimePeriod = 50ms.
- ValidCountLimit = 2
- FailCountLimit = 3

In the above example, the UA must set Refresh once every 50ms for the timer to be satisfied. There are two inner states associated with the watchdog. They are WatchdogExpired and WatchdogNormal. The CDS should evaluate the refresh and timer counts once every 50ms as follows:

- When the state is W makes the ShowIfFailIdent widget visible. Every 50ms (the TimePeriod) the CDS checks to see if the refresh has been updated by the UA in the last 50ms. If the refresh is updated for 2 consecutive cycles (100ms) then the CDS transitions to the WatchdogNormal state.
- When the state is WatchdogNormal. In the WatchdogNormal state the CDS evaluates the other widgets for display. Every 50ms (the TimePeriod) the CDS checks to see if Refresh has been updated by the UA in the last 50ms. If the refresh is not updated for 3 consecutive cycles (150ms) then the CDS transitions to the WatchdogExpired state.watchdogExpired – In the WatchdogExpired state the CDS.

The figure below shows a state transition diagram for the Watchdog Container. With TimePeriod equal to 50ms, this State Transition Diagram would be evaluated once every 50ms by the CDS. For the labels on the transitions the condition above the line determines whether the transition is taken and the statement under the line is the action taken if the transition occurs.



Restriction:
N/A

Figure 3.5.3-2 – Watchdog State

Table 3.5.3-1 – WatchdogContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_WATCHDOG_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Specific parameters		
TimePeriod	D	The period in ms between updates of timer. Note: The values allowed for this parameter depend on the CDS supplier.
ValidCountLimit	D	The number of periods that the refresh must be updated before the Watchdog is satisfied.
FailCountLimit	D	The number of periods that the refresh is not updated before the Watchdog expires.
Refresh	R	Set by the UA to satisfy the Watchdog. There is no significance to the value passed to the Refresh parameter.
ShowIfFailIdent	D	The ID of the child widget to be evaluated for display when the timer expires. A value of zero indicates that nothing is to be displayed when the timer expires.
ShowFail	DR	ShowFail can be set to true by the UA to show the failed state of the widget. This has the same result as failing to update the Refresh.

Table 3.5.3-2 – WatchdogContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_WATCHDOG_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	16	
TimePeriod	ushort	16	
ValidCountLimit	ushort	16	
FailCountLimit	ushort	16	
Refresh	uchar	8	
UnusedPad	N/A	8	
ShowIfFailIdent	ushort	16	
ShowFail	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	

Table 3.5.3-3 – WatchdogContainer Event Structures: A661_EVT_WATCHDOG_EXPIRED

CreateParameter Buffer	Type	Size (bits)	Value/Description
EventIdent	Ushort	16	A661_EVT_WATCHDOG_EXPIRED
UnusedPad	N/A	16	

Table 3.5.3-4 – WatchdogContainer Event Structures: A661_EVT_WATCHDOG_NORMAL

CreateParameter Buffer	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_WATCHDOG_NORMAL
UnusedPad	N/A	16	

Table 3.5.3-5 – WatchdogContainer Runtime Modifiable Parameters

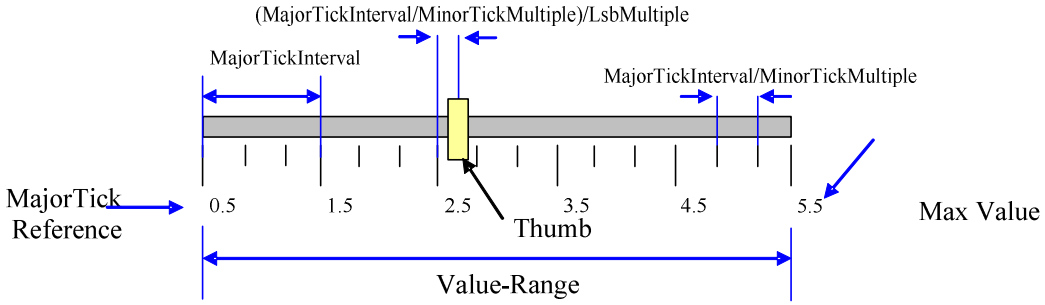
Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParamterStructure	Type of Structure Used (Refer to 4.5.4.5)
Refresh	uchar	8	A661_ REFRESH	A661_ParameterStructure_1Byte
ShowFail	uchar	8	A661_ SHOW_FAIL	A661_ParameterStructure_1Byte

3.5.4 Slider

Categories:

- Graphical Representation
- Interactive

Description:



A Slider allows the crewmember to select a value between the range of MIN_VALUE and MAX VALUE. The Slider can be displayed in either the horizontal or vertical axis.

The mapping of the value-range to the slider depends on the orientation, for example when the slider is horizontal the value-range can either be LEFT TO RIGHT or RIGHT TO LEFT. The Orientation specifies direction of increasing values.

When the UA sets the Value to something that is not a multiple of the LSB value the behavior is implementation dependent

The event is reported on upon selection by a crewmember with a “click” or keyboard selection. The CDS would then move the thumb to the click position.

In the diagram above, the following values are used:

- MajorTickReference = 0.5
- MajorTickInterval = 1.0
- MinorTickMultiple = 3
- LsbMultiple = 2

Restriction:
MAX_VALUE must be greater than the MIN_VALUE.

Table 3.5.4-1 – Slider Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SLIDER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MinValue	D	Minimum value of the scroll bar
MaxValue	D	Maximum value of the scroll bar
Value	DR	Current value of the Slider
MajorTickInterval	D	Value between each MAJOR tick mark. MajorTickInterval can not be zero.
ShowMajorLabels	D	If TRUE the values for each MAJOR tick are shown.
Alignment	D	Specifies where the ticks and labels are drawn with relation to the slider. A661_LEFT A661_RIGHT A661_CENTER A661_TOP A661_BOTTOM
Orientation	D	Specifies the orientation of the Slider either vertical or horizontal. It also specifies which way the value-range of the slider is oriented. A661_BOTTOM_TO_TOP - Vertical A661_TOP_TO_BOTTOM - Vertical A661_LEFT_TO_RIGHT - Horizontal A661_RIGHT_TO_LEFT - Horizontal
LsbMultiple	D	This specifies the number of intervals between discrete positions for the thumb within the Minor Tick.
MinorTickMultiple	D	Specifies the number of Minor Tick intervals between two Major Ticks.
MajorTickReference	D	Holds the value where the major tick starts.

Parameters	Change	Description
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

Table 3.5.4-2 – Slider Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SLIDER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MinValue	float	32	
MaxValue	float	32	
Value	float	32	
MajorTickInterval	float	32	
ShowMajorLabels	uchar	8	A661_TRUE A661_FALSE
Orientation	uchar	8	A661_BOTTOM_TO_TOP A661_TOP_TO_BOTTOM A661_LEFT_TO_RIGHT A661_RIGHT_TO_LEFT
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER A661_TOP A661_BOTTOM
UnusedPad	N/A	8	
LsbMultiple	ushort	16	
MinorTickMultiple	ushort	16	
MajorTickReference	float	32	

Table 3.5.4-3 – Slider Event Structures Tables: A661_EVT_VALUE_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_VALUE_CHANGE
UnusedPad	N/A	16	0
Value	float	32	Holds the current value of the Slider

Available SetParameter identifiers and associated data structure are defined in Table 3.5.4-4.

Table 3.5.4-4 – Slider Runtime Modifiable Parameters Table

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
Value	float	32	A661_VALUE	A661_ParameterStructure_4Bytes
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.5.5 PictureAnimated

Categories:
Graphical representation

Description:
A PictureAnimated is a reference to a set of images available in the CDS. By displaying this set of pictures successively at a frequency defined as a parameter of the widget, the CDS performs an animation. It is possible to define if the animation is performed only once then stopped or if it is an infinite loop. The UA can stop the loop animation. The animation always stops on the first picture of the list.

Restriction:
N/A

Table 3.5.5-1 – PictureAnimated Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PICTURE_ANIMATED
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
Specific parameters		
IndexOfFrequency	D	Index of the frequency defined in CDS (OEM dependent)
LoopFlag	D	Flag for loop animation: FALSE: animation is played once then stops TRUE: loop animation.
AnimationFlag	DR	TRUE: the animation is played FALSE: the animation continues until it reaches the first picture of the array. Then it stops.
NumberOfPictures	D	The number of picture references to store
PictureArray	D	Array of pictures references stored in the CDS. The definition order set the sequence order.

PictureAnimated Creation Structure is defined in Table 3.5.5-2.

Table 3.5.5-2 – PictureAnimated Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE_ANIMATED
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	uchar	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NumberOfPictures	uchar	16	
IndexOfFrequency	uchar	8	
LoopFlag	uchar	8	A661_FALSE A661_TRUE
AnimationFlag	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
PictureArray	{ushort}+	16* NumberOfPictures + PAD	Array of pictures references Followed by zero or two extra NULL for alignment on 32 bits.

The PictureAnimated widget does not send any event.

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.5.5-3 – PictureAnimated Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
AnimationFlag	uchar	8	A661_ANIMATION_FLAG	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes

3.5.6 SymbolAnimated

Categories:
Graphical representation

Description
The SymbolAnimated widget is similar to the Symbol widget. It places vector symbols predefined in the CDS at a specified location on the display. However, the Symbol widget is static unless the controlling application changes widget parameters, while the SymbolAnimated widget is animated by providing the ability to automatically display a sequence of symbol references, rotation angles and relative movements.

NumberOfSymbols specifies how many elements are specified in the creation structure for each of the following arrays: a symbol reference array, a rotation angle array, and arrays for both an x and a y component of relative movement (relative to the widgets overall PosX/PosY position). The combination of these arrays allows the

definition of animation sequences in which the symbol displayed by the CDS, its orientation and position relative to the widget's origin can change over the course of time.

IndexOfFrequency is interpreted according to CDS-specific rules. This widget attribute controls the time between adjacent cycles of the animation sequence when the animation is played.

AnimationType allows the controlling application to start and stop the animation. When the widget becomes visible and AnimationType is set to A661_DONT_RUN, the first symbol reference, orientation, and movement parameters are used to display the symbol. If AnimationType is A661_RUN_ONCE or A661_RUN when the widget is set to visible, the animation begins. A661_RUN_ONCE will start the animation and have the CDS stop it automatically once the animation sequence has been completely displayed – to start over, AnimationType must be set to A661_RUN_ONCE again. To show the animation continuously, AnimationType should be set to A661_RUN and remain this value until the animation should stop, at which time the value should be reset to A661_DONT_RUN.

LoopType is one of the following:

- A661_LOOP_FORWARD:
  The sequence is played from the first to the last frame. When the animation is stopped, the last frame remains visible
- A661_LOOP_FORWARD_AND_RESET:
  The sequence is played from the first to the last frame. When the animation is stopped, the first frame remains visible
- A661_LOOP_FORWARD_AND_BLANK:
  While the animation is running, the sequence is played from the first to the last frame. When the animation is not running, nothing is visible
- A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK:
  While the animation is running, the sequence is played from the first to the last frame and then reverses direction until the first frame is reached again. When the animation is not running, nothing is visible

Restriction:
N/A

Table 3.5.6-1 – SymbolAnimated Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SYMBOL_ANIMATED
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside the CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
Specific parameters		
MotionAllowed	D	Capability to change PosX, PosY at run time
ColorIndex	DR	Color index, used if StyleSet allows color to be set
IndexOfFrequency	D	Index of the frequency defined in the CDS (Implementation dependent)

Parameters	Change	Description
LoopType	D	Type of looping (see description above)
AnimationType	DR	Type of animation (see description above)
NumberOfSymbols	D	The number of symbol references used for this animation.
SymbolArray	D	Array of symbol references
OrientationArray	D	Array of symbol orientations
MovementXArray	D	Array of horizontal component of movement
MovementYArray	D	Array of vertical component of movement

Table 3.5.6-2 – SymbolAnimated Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_SYMBOL_ANIMATED
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
MotionAllowed	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	
LoopType	uchar	8	A661_LOOP_FORWARD A661_LOOP_FORWARD_AND_RESET A661_LOOP_FORWARD AND_BLANK A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK
NumberOfSymbols	ushort	16	
IndexOfFrequency	uchar	8	
AnimationType	uchar	8	A661_DONT_RUN A661_RUN A661_RUN_ONCE
SymbolArray	{ushort}+	16 * NumberOfSymbols + PAD	Array of symbol references, padded if necessary to reach a 32 bit boundary
OrientationArray	{fr(180)}+	32 * NumberOfSymbols	Array of rotation angles
MovementXArray	{long}+	32 * NumberOfSymbols	Array of horizontal movements.
MovementYArray	{long}+	32 * NumberOfSymbols	Array of vertical movements.

Table 3.5.6-3 – SymbolAnimated Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParamterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
PosX	long	32	A661_POS_X	A661_ParameterStructure_4Bytes
PosY	long	32	A661_POS_Y	A661_ParameterStructure_4Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_ParameterStructure_8Bytes
ColorIndex	uchar	8	A661_COLOR_INDEX	A661_ParameterStructure_2Bytes
Animation**Type**	uchar	8	A661_ANIMATION_**TYPE**	A661_ParameterStructure_1Byte

3.5.7 SelectionListButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The SelectionListButton is the same as the ComboBox except that a constant LabelString is displayed instead of the selected value.

The SelectionListButton allows a crew member to select one entry within a list. A fixed LabelString is displayed in the SelectionListButton area. The number of the current selected entry is held in the SelectedEntry parameter. The complete list of possible Entries is held in a string array (parameter EntryList). The list is displayed upon crew member selection, for example, click on the arrow button associated with the Selected Entry.

Note that SelectingAreaHeight and the SelectingAreaWidth represent the Y and X Size of the PopUp part of the SelectionListButton .

OpeningMode of the SelectionListButton determines how the SelectionListButton opens.

The pop-up part of the SelectionListButton is displayed on top of its containing window and is affected by the clipping area of its containing window.

Restriction:
N/A

SelectionListButton Parameters are defined in Table 3.5.7 -1.

Table 3.5.7 -1 – SelectionListButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SELECTION_LIST_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the SelectionListButton (in the closed mode)
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaHeight	D	Size of the area available to display the entry list
SelectingAreaWidth	D	Size of the area available to display the entry list
LabelString	D	Text permanently displayed in the SelectionListButton label area.
OpeningMode	D	Way of combo opening: UP CENTERED DOWN
MaxStringLength	D	Maximum string length for **each** entr**y in** the list.
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be lower than MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list.
OpeningEntry	DR	Entry number which is ensured to be visible when the SelectionListButton is opened. Opening entry is in the range [0; NumberOfEntries] OpeningEntry will be set to 0, if not used.
EntryList [MaxEntryNumber]	DR	String array holding the list of entries.
EntryValidation	**R**	**Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.** **A661_FALSE** **A661_TRUE**

SelectionListButton Creation Structure is defined in Table 3.5.7-2.

Table 3.5.7-2 – SelectionListButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SELECTION_LIST_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningEntry	ushort	16	
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0
LabelString	string	8 x string length+ PAD	The string is followed by zero, one, two or three NULL character(s) to be 32 bits aligned
EntryList [NumberOfEntries]	{string}+	{32}+	Each string terminating NULL is used as string separator. The complete string list is followed by zero, one, two or three NULL character(s) to be 32 bits aligned

The specific event sent by the SelectionListButton to the owner application is defined by Table 3.5.7 -3.

Table 3.5.7-3 – SelectionListButton Event Structures: A661_EVT_SEL_ENTRY_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_SEL_ENTRY_CHANGE
EntryNumber	ushort	16	Number of the entry chosen by the crew member

Available SetParameter identifiers and associated data structure are defined in Table 3.5.7 -4.

SelectionListButton Runtime Modifiable Parameters are defined in Table 3.5.7 -4.

Table 3.5.7-4 – SelectionListButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	16	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
OpeningEntry	ushort	16	A661_OPENING_ENTRY	A661_ParameterStructure_2Bytes
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES	A661_ParameterStructure_2Bytes
EntryList [NumberOfEntries]	{string}+	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray
EntryValidation	**uchar**	**8**	**A661_ENTRY_VALID**	**A661_ParameterStructure_1Byte**

3.6 Widget Extension (Supplement 3)

3.6.1 EditBoxNumericBCD

Categories:

- **Graphical representation**
- **Interactive**
- **Text string**

Description:

The EditBoxNumericBCD is a customized numeric edit box allowing to edit a complex numeric value that is not in 10 base (such as a time). The principle of this widget is to use several fields for the value, each field being a part of the value (e.g. For time = 23h59 min, there is one field for hour from 0 to 23 and one field for minutes from 0 to 59)).A crew member can modify this value using its input devices. As it is a numeric value, CDS is able to increment itself the value. The widget can receive a number of increment or a numeric key value.

The idea is to use distinct fields of "value" 64 bits word. The size of each field is defined in the widget. Each 4 bits part of "value" parameter is used to code a digit (in BCD format), of the value to be displayed. The first 4 bits field is used to code the sign (0xF=-1 / 0x1=+1).

Example:

0x**F**123456789012345:

sign: odd (**F**)

1st digit = 1

2nd digit = 2

…

15th digit = 5

The number of fields is defined by the parameter NumberOfFields.

The table SizeOfFields[NumberOfFields] allows to define the size of each field in digits. Note that the format limits the global size of parameter value to 15 digits. It is not necessary to use all the 15 digits. It is not allowed to define a field size of 0.

Examples:

	Example 1	Example 2	Example 3
NumberOfFields	2	4	2
SizeOfFields	3; 2	3; 2; 2; 4	2; 2
Value	0x1123450000000000	0xF179595912340000	0x1094500000000000
decoding	Sign: + Field1: 123 Field2: 45	Sign: - Field1: 179 Field2: 59 Field3: 59 Field 4: 1234	Sign: + Field1: 09 Field2: 45

The widget parameters MinFieldsValue and MaxFieldsValue define the min and max boundaries of each field in ABSOLUTE VALUE (always positive). The widget parameters MinFieldsValues and MaxFieldsValues use the same coding of 'value' parameter but the sign is not interpreted.

MinValue and MaxValue parameters allow to define the min and max of the global signed value.

MinValue and MaxValue parameters use the same coding of 'value' parameter

Examples:

	Example 1	Example 2
NumberOfFields	4	2
SizeOfFields	3; 2; 2; 4	2; 2
MinFieldsValues	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 Field2: 0 => 0x000000000000000
MaxFieldsValues	Field1: 180 Field2: 59 Field3: 59 Field 4: 9999 => 0x0180595999990000	Field1: 23 Field2: 59 => 0x0235900000000000
MinValue	0xF179595999990000 - 17959599999	0x1000000000000000 0
MaxValue	0x1180000000000000 + 18000000000	0x1235900000000000 + 2359

The widget parameters TicsFine and TicsCoarse use the same coding than value parameter.

The parameters PositiveString and NegativeString define a string for positive and negative values. The character '+' in parameter FormatString indicates the position of parameter PositiveString or NegativeString for display.

Examples:

	Example 1	Example 2	Example 3
NumberOfFields	4	4	2
SizeOfFileds	3; 2; 2; 4	3; 2; 2; 4	2; 2
PositiveString	EAST	null	RIGHT
NegativeString	WEST	null	LEFT
FormatString	###°##'##.####"+	+__#°##'##.#___"	##.## +
MinFieldsValue	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 = 0x00 Field2: 0 = 0x00 => 0x0000000000000000
MaxFieldsValue	Field1: 180 Field2: 59 Field3: 59 Field4: 9999 => 0x0180595999990000	Field1: 360 Field2: 59 Field3: 59 Field4: 9999 => 0x0360595999990000	Field1: 90 Field2: 99 => 0x0909900000000000
MinValue	-179 59 59 9999 => 0xF179595999990000	0 00 00 0000 => 0x1000000000000000	-90 00 => -9000 => 0xF900000000000000
MaxValue	180 00 00 0000 => 18000000000 => 0x1180000000000000	359 59 59 9999 => 35959599999 => 0x1359595999990000	90 00 => 9000 => 0x1900000000000000
TicsFine	Field1: 0 Field2: 0 Field3: 0 Field4: 50 => 0x0000000000500000	Field1: 0 Field2: 0 Field3: 0 Field4: 50 => 0x0000000000500000	Field1: 0 Field2: 1 => 0x0000100000000000
TicsCoarse	Field1: 0 Field2: 0 Field3: 0 Field4: 1000 => 0x0000000010000000	Field1: 0 Field2: 0 Field3: 0 Field4: 500 => 0x0000000005000000	Field1: 1 Field2: 0 => 0x0010000000000000
Value (example)	Sign: + Field1: 45 Field2: 5 Field3: 10 Field4: 4200 0x1045051042000000	Sign: + Field1: 45 Field2: 5 Field3: 10 Field4: 4200 0x1045051042000000	Sign: Field1: 79 Field2: 79 0xF797900000000000
Display	045°05'10.4200"EAST	45°05'10.42"	79.79 LEFT

Some of the parameters of the EditBoxNumericBCD have to be consistent:

- Each field of the Value must belong to [MinfieldValue, MaxfieldValue]
- The Value must belong to [MinValue, MaxValue] and be consistent with FormatString
- MinValue must be less than MaxValue
- MinFieldValue must be less than MaxFieldValue
- The length of the string edited with the KCCU and the length of the FormatString shall be less than MaxFormatStringLength

- During the run-time phase, if the UA sets any field of parameter "Value" with a value inferior to MinFieldValue or greater than MaxFieldValue, the CDS shall notify an error. In the same way, if during the run-time phase, the UA sets the parameter Value with a value inferior to MinValue or greater than MaxValue, the CDS shall notify an error. However, the values entered by the crew member are not checked by the CDS.

When the EditBoxNumericBCD is in Edit Mode, the CDS may report all modification done on the edited value and the final confirmed value, or only report the confirmed value (after a crew member validation). This option is setable by the UA through the "ReportAllChanges" parameter. If after having entered a value, the crewmember finally aborts the edition, the CDS shall send a specific event to the User Application with the former validated Value as parameter of the event.

Upon event notification A661_STRING_CHANGE & A661_STRING_CONFIRMED (respectively A661_STRING_CHANGE_ABORTED), the string value returned to the UA shall be the string currently displayed in the edit area (respectively the 'value' parameter converted in string according to formatString)

EditBoxNumericBCD Parameters are defined in Table 3.6.1-1.

Table 3.6.1-1 – EditBoxNumeric Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_NUMERIC_BCD
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	D	Automatic motion of the focus on widget specified in NextFocusedWidget parameter.
Specific parameters		
Value	DR	Value to be interpreted (format Binary Coding Decimal). Each half-byte will be interpreted as a digit to be displayed in the final string except the most significant one which will be used for the sign (F for a negative value and 1 for a positive value). Example: 0xF123456789012345 - negative sign (F) - 1st digit = 1 - 2nd digit = 2 - … - 15th digit = 5
NumberOfFields	D	Number Of Fields in the Value
SizeOfFields [NumberOfFields]	D	Number of digits of each field of the Value

<table>
<tr><th>Parameters</th><th>Change</th><th>Description</th></tr>
<tr><td>MinFieldsValues</td><td>D</td><td>Minimum value of each field (in absolute value)</td></tr>
<tr><td>MaxFieldsValues</td><td>D</td><td>Maximum value of each field (in absolute value)</td></tr>
<tr><td>MinValue</td><td>D</td><td>Minimum value of the parameter Value</td></tr>
<tr><td>MaxValue</td><td>D</td><td>Maximum value of the parameter Value</td></tr>
<tr><td>TicsCoarse</td><td>DR</td><td>Coarse increment step for modification of the value with main wheel</td></tr>
<tr><td>TicsFine</td><td>DR</td><td>Fine increment step for modification of the value with secondary wheel</td></tr>
<tr><td>FormatString</td><td>DR</td><td>String describing the format of the numeric field. The string is composed of any authorized characters. Some of them will be interpreted to format the value.
'_' and '#' are used to position the different digits within the string (the first '_' or '#' indicates the position of the first digit and so on …).

'_' : is used to remove leading and trailing 0 within the same field. If the digit is equal to 0, it indicates that the corresponding digit is replaced by no character, or it is displayed when there is a digit on the left and on the right of the 0 within the same field. In other cases ([1-9]), it indicates that the corresponding digit will be displayed as it is defined in the value. Refer to Figure 3.6.1-2 for more information.

'#' : indicates that the corresponding digit will be displayed as it is defined in the value.
'+': defines the position of the PositiveString/NegativeString in the final string. Note that if this sign character is not present in the FormatString, there will be no difference between the display of a positive or a negative value.
'*': Any other characters (any except one of '+', '_', '#') will be displayed

Examples:
<table>
<tr><th>NumberOfFields / SizeOfFields</th><th>PosString / NegString FormatString</th><th>Value</th><th>Displayed string</th></tr>
<tr><td>2 / 4; 1</td><td>+ / -
+___#.#</td><td>0xF0123400</td><td>- 123.4</td></tr>
<tr><td>4 / 3; 2; 2; 4</td><td>EAST / WEST
###°##'##.####"
+</td><td>0xF17959591
3240000</td><td>179°59'59.1234"W
EST</td></tr>
<tr><td>3 / 3; 2; 2</td><td>null / -
+###°##'##"</td><td>0x10054630</td><td>005°46'30"</td></tr>
<tr><td>2 / 2; 2</td><td>RIGHT / LEFT
_#.#_ +</td><td>0x10510000</td><td>5.1 RIGHT</td></tr>
<tr><td>2 / 3; 2</td><td>+ / -
###x##</td><td>0x12501500</td><td>250x15</td></tr>
</table></td></tr>
<tr><td>MaxFormatStringLength</td><td>D</td><td>Maximum size of the FormatString.</td></tr>
<tr><td>PositiveStringLength</td><td>D</td><td>Length of the PositiveString</td></tr>
<tr><td>NegativeStringLength</td><td>D</td><td>Length of the NegativeString</td></tr>
<tr><td>PositiveString</td><td>D</td><td>String to be used for a positive value</td></tr>
<tr><td>NegativeString</td><td>D</td><td>String to be used for a negative value</td></tr>
<tr><td>Alignment</td><td>D</td><td>Justification of the label text within the edit box area
CENTER</td></tr>
</table>

Parameters	Change	Description
		LEFT RIGHT
ReportAllChanges	D	A661_EDB_CHANGE_CONFIRMED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) A661_EDB_ALL_CHANGE CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) A661_EDB_OPEN_CLOSE CDS will report the edit mode opening A661_EVT_EDITBOX_OPENED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
MaxLegendStringLength	D	Max string size for the legend (MaxLegendStringLength > 0)
LegendAreaSizeX	D	The X dimension (size) of the legend.
LegendString	DR	Legend associated to the numeric value
LegendPosition	D	Position of the legend in comparison with the numeric value: LEFT, RIGHT, TOP, BOTTOM
LegendRemoved	D	The flag defining if the legend shall be removed on entry in the editing mode.
NumericKeyFlag	D	Ability to change the value with the numerical key TRUE FALSE
CyclicFlag	D	Possibility for cyclic modification of the value TRUE FALSE

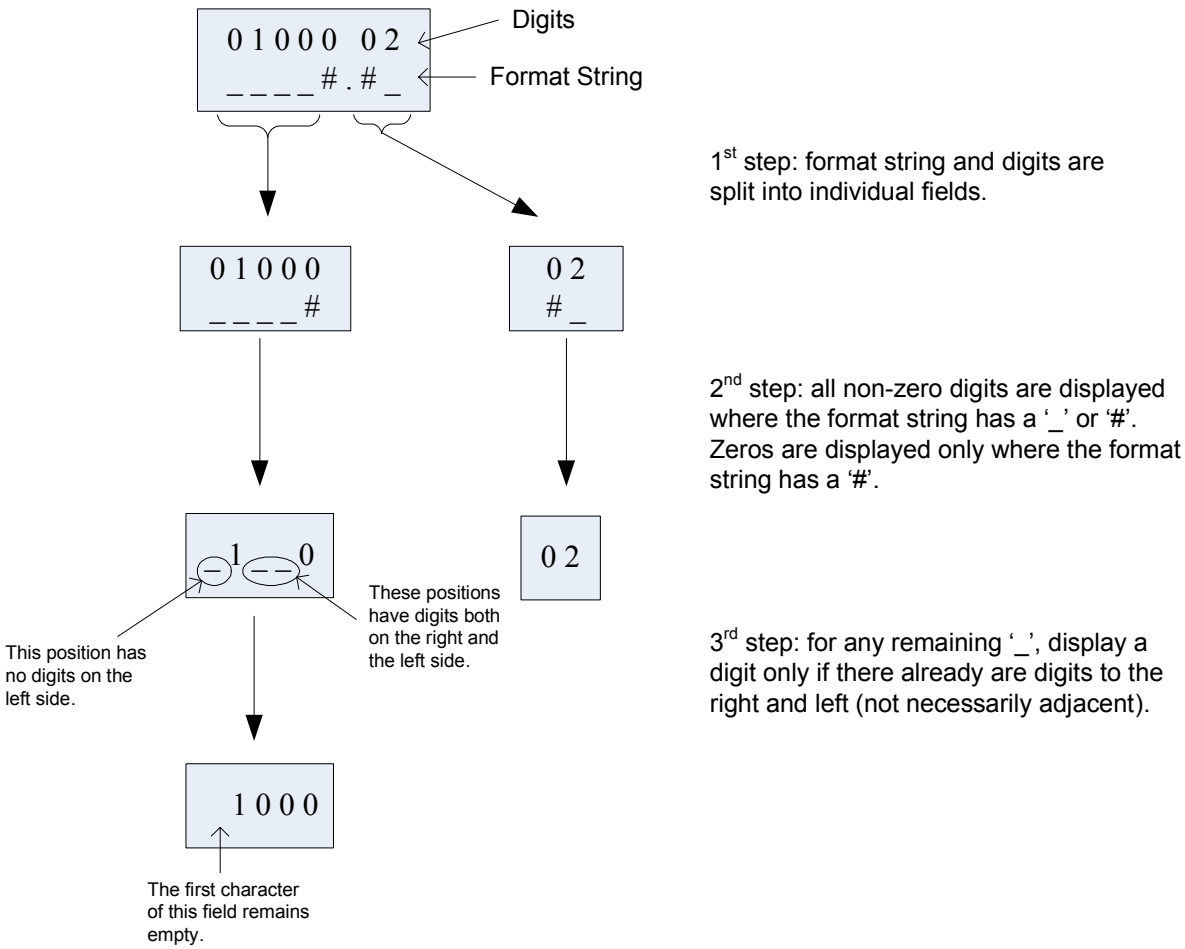


Figure 3.6.1-2 – Format String example of '_' character

EditBoxNumeric Creation Structure is defined in Table 3.6.1-2.

Table 3.6.1-2 – EditBoxNumeric Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range when necessary
WidgetType	ushort	16	A661_EDIT_BOX_NUMERIC_BCD
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
LegendAreaSizeX	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
Value	BCD	64	Comment : Each 4 bits will be interpreted as a digit to be displayed in the final string except the most significant

CreateParameterBuffer	Type	Size (bits)	Value / Range when necessary
			one which will be used for the sign (F for a negative value and 1 for a positive value).
MinValue	BCD	64	Comment : Same type as value
MaxValue	BCD	64	Comment : Same type as value
MinFieldsValues	BCD	64	Comment : Same type as value but the sign will not be interpreted.
MaxFieldsValues	BCD	64	Comment : Same type as value but the sign will not be interpreted.
TicsCoarse	BCD	64	Comment : Same type as value
TicsFine	BCD	64	Comment : Same type as value
NumberOfFields	uchar	8	
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
LegendPosition	uchar	8	A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
LegendRemoved	uchar	8	A661_FALSE A661_TRUE
MaxFormatStringLength	uchar	8	
MaxLegendStringLength	uchar	8	
PositiveStringLength	uchar	8	
NegativeStringLength	uchar	8	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_FALSE A661_TRUE
NumericKeyFlag	uchar	8	A661_FALSE A661_TRUE
CyclicFlag	uchar	8	A661_FALSE A661_TRUE
SizeOfFields [NumberOfFields]	uchar	8 * NbOf Fields + PAD	Followed by zero, one, two or three NULL for alignment on 32 bits
FormatString	string	8 * string length	
LegendString	string	8 * string length	
PositiveString	string	8 * string length	
NegativeString	string	8 * string length + PAD	Followed by zero, one, two or three extra NULL for alignment on 32 bits.

EditBoxNumericBCD Event Structures Table

During edition of an EditBoxNumericBCD, it shall be possible for the pilot to enter values containing alphanumeric character (for example, the value and the legend). It is the responsibility of the UA to analyze the entered value and format it correctly in the widgets. As a consequence of this need, the events associated to the EditBoxNumericBCD take a string value as parameters.

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE_ABORTED are defined in Table 3.6.1-3.

Table 3.6.1-3 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE_ABORTED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE_ABORTED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE are defined in Table 3.6.1-4.

Table 3.6.1-4 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CHANGE
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CONFIRMED are defined in Table 3.6.1-5.

Table 3.6.1-5 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STRING_CONFIRMED
StringLength	ushort	16	
String	string	{32}+	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumericBCD Event Structures: A661_EVT_EDITBOX_OPENED are defined in Table 3.6.1-6.

Table 3.6.1-6 – EditBoxNumericBCD Event Structures: A661_EVT_EDITBOX_OPENED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_EDITBOX_OPENED
UnusedPad	ushort	16	0

The CDS shall notify an event to the User Application when the EditBox widget goes to Editing state. As a consequence, the CDS shall also notify the UA when the EditBox widget exits the Editing State without any validation, using the event A661_STRING_CHANGE_ABORTED.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.1-7.

Table 3.6.1-7 – EditBoxNumericBCD Runtime Modifiable Parameters

	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
EntryValidation	uchar	8	A661_ENTRY_VALID	A661_ParameterStructure_1Byte
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes
Value	BCD	64	A661_VALUE	A661_ParameterStructure_8bytes
Tics coarse	BCD	64	A661_TICS_COARSE	A661_ParameterStructure_8bytes
Tics fine	BCD	64	A661_TICS_FINE	A661_ParameterStructure_8bytes
LegendString	string	{32}+	A661_STRING	A661_ParameterStructure_String
FormatString	String	{32}+	A661_FORMAT_STRING	A661_ParameterStructure_String

3.6.2 CursorRef

Categories:

Utility

Description:
The CursorRef widget defines a non-visible reference point that the Cursor can be moved to using the A661_REQ_CURSOR_ON_WIDGET command.

The ability for a UA to request that the cursor be placed on an item in its layer is important for certain Human Engineering considerations. These include situations where it is important to place the cursor so that it is not on a widget (i.e. at a point on the screen), to avoid inadvertent selection, or in cases where the widget that the cursor is being placed upon is complex and the cursor needs to be placed on a particular sub-part of that widget.

The widget has no graphics associated with it. The values of PosX, PosY defines its location.

The CursorRef can be the normal child of a container, layer, MapHorz_Source or MapVert_Source. Depending upon this parent, the widget parameters PosX and PosY are interpreted as either screen co-ordinates (1/100mm) or the co-ordinates of the associated map source. Values of PosX, PosY must be set appropriately to this co-ordinate system for the CursorRef to be positioned at the intended point.

Restriction:

None

CursorRef Parameters are defined in Table 3.6.2-1.

Table 3.6.2-1-1 – CursorRef Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CURSOR_REF
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
PosX	DR	The X / Lat / Range position of the widget reference point. First coordinate of reference point
PosY	DR	The Y / Long/ Bearing position of the widget reference point. Second coordinate of reference point

CursorRef Creation Structure is defined in Table 3.6.2-2.

Table 3.6.2-2 – CursorRef Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CURSOR_REF
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	16	0
PosX	long/ scaled Integer	32	If parent is a MapHorz_Source then the LSB and units are defined by the MapHorz_Source. If parent is not a MapHorz_Source then the units are 1/100mm.
PosY	long/ scaled integer	32	If parent is a MapHorz_Source then the LSB and units are defined by the MapHorz_Source. If parent is not a MapHorz_Source then the units are 1/100mm.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.2-3.

Table 3.6.2-3 – CursorRef Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
PosX	Long / scaled integer	32	A661_POS_X	A661_Parameter_structure_4Bytes
PosY	Long / scaled integer	32	A661_POS_Y	A661_Parameter_structure_4Bytes
PosX PosY	long x 2 / scaled integer x 2	32 x 2	A661_POS_XY	A661_Parameter_structure_8Bytes

3.6.3 CursorOver

Categories:

Interactive
Graphical Representation

Description:
The CursorOver widget allows the UA to be notified as soon as a Cursor enters or exits the active area of the widget. The events generated do not require a cursor select action, they are generated when the X,Y position of the cursor intersects with the active area of the widget.

Unlike the ActiveArea and CursorPosOverlays widgets, no additional events are created by the CursorOver widget when the widget is clicked-on.

The events that are generated when the cursor enters the active area of the widget can be configured as follows, using the PositionReport Mode parameter.

- **Report All: Events are sent when the cursor enters, exits or while over the active area.**

- **On Transition: Events are sent only when the cursor enters or exits the active area.**

The event notification reports the X,Y position of the Cursor with respect to the origin of the CursorOver widget.

The StyleSet may be used to define alternate cursor representations.

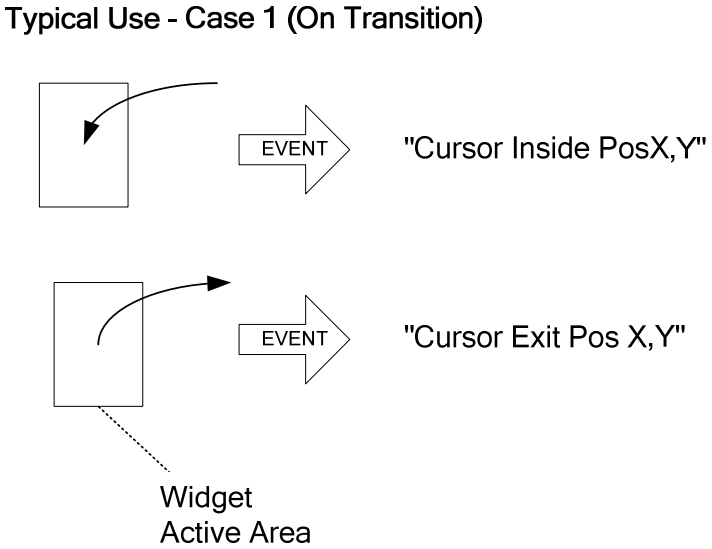
While the Cursor is over the widget an alternative CDS defined cursor or other graphics can be displayed if desired, as defined by the widget StyleSet.

The EventOrigin parameter contained in the A661_NOTIFY_WIDGET_EVENT block defines which cursor has entered the active area (see section 4.5.4.2).

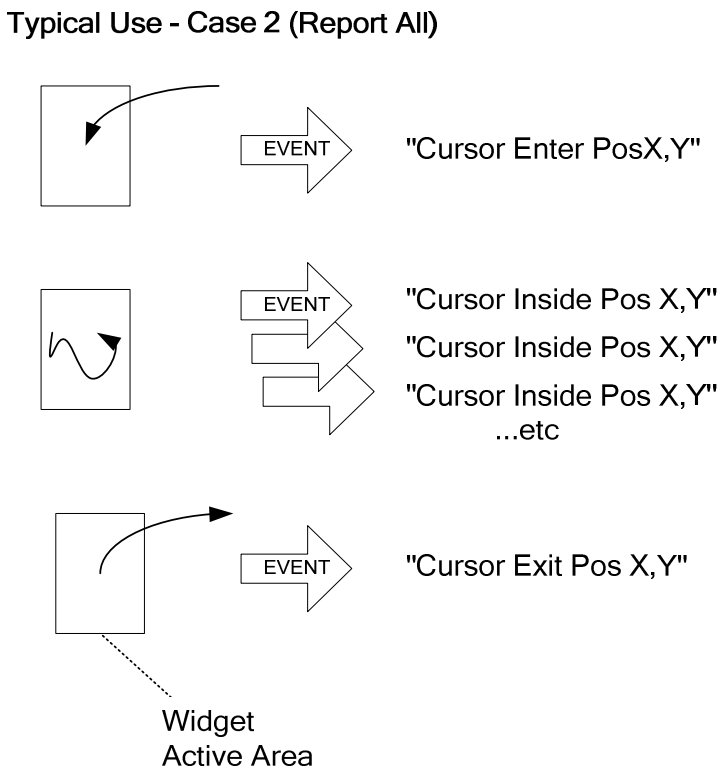
CursorOver widgets that are drawn on top of other interactive widget do not prevent the widget underneath from generating events. Under these conditions the CursorOver widget and the interactive widget underneath will still be able to generated events. This is one of the noted exceptions to the rule defined in section 2.3.5.2 for overlapping widgets.

There are many possible uses for this widget for detecting whether a cursor is within a certain region on the screen. Some example cases are shown below.

Example case 1: By configuring the widget to report on transition, events are only sent to the UA when the cursor passes outside to inside and inside to outside.



Example case 2: By configuring the widget to report all, events are sent to the UA when the cursor passes from outside to inside; continuously while the cursor is inside; and from inside to outside.



Restriction:

None

CursorOver Parameters are defined in Table 3.6.3-1.

Table 3.6.3-1 – CursorOver Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CURSOR_OVER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The width of the widget.
SizeY	DR	The height of the widget.
PositionReport Mode	D	Defines when events are generated from the widget, when the Cursor is over the widget active area. A661_REPORT_ON_TRANSITION - A661_EVT_CURSOR_ENTER, sent once when cursor enters the active area. - A661_EVT_CURSOR_EXIT, sent once when cursor exits the active area. A661_REPORT_ALL - A661_EVT_CURSOR_ENTER, sent once when cursor enters the active area.

Parameters	Change	Description
		- A661_EVT_CURSOR_EXIT, sent once when cursor exits the active area. - A661_EVT_CURSOR_INSIDE, sent continuously while the cursor remains inside the active area.
StyleSet	DR	Defines optional characteristics of the cursor or other related graphics as defined by the CDS implementation.

CursorOver Creation Structure is defined in Table 3.6.3-2.

Table 3.6.3-2 – CursorOver Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CURSOR_OVER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
Unused Pad	N/A	16	
PositionReportMode	uchar	8	A661_REPORT_ON_TRANSITION A661_REPORT_ALL
Unused pad	N/A	24	

Note: The Visible and Enable parameters are in a different order to the normal convention.

The specific events sent by the CursorOver to the owner application are defined in Table 3.6.3-3, -4, and -5.

Table 3.6.3-3 – CursorOver Event Structures: A661_EVT_CURSOR_ENTER

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT _CURSOR_ENTER
UnusedPad	N/A	16	
RelPosX	long	32	Relative X position of Cursor as an offset to the widgets origin.
RelPoxY	long	32	Relative Y position of Cursor as an offset to the widgets origin.

Table 3.6.3-4 – CursorOver Event Structures: A661_EVT_CURSOR_INSIDE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT _CURSOR_INSIDE
UnusedPad	N/A	16	
RelPosX	long	32	Relative X position of Cursor as an offset to the widgets origin.
RelPoxY	long	32	Relative Y position of Cursor as an offset to the widgets origin.

Table 3.6.3-5 – CursorOver Event Structures: A661_EVT_CURSOR_EXIT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_EXIT
UnusedPad	N/A	16	
RelPosX	long	32	Relative X position of Cursor as an offset to the widgets origin.
RelPoxY	long	32	Relative Y position of Cursor as an offset to the widgets origin.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.3-6.

Table 3.6.3-6 – CursorOver Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Visible	uchar	8	A661_VISIBLE	A661_Parameter_structure_1Byte
Enable	uchar	8	A661_ENABLE	A661_Parameter_structure_1Byte
PosX	long	32	A661_POS_X	A661_Parameter_structure_4Bytes
PosY	long	32	A661_POS_Y	A661_Parameter_structure_4Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_Parameter_structure_8Bytes
SizeX	ulong	32	A661_SIZE_X	A661_Parameter_structure_4Bytes
SizeY	ulong	32	A661_SIZE_Y	A661_Parameter_structure_4Bytes
SizeXY	long x 2	32 x 2	A661_SIZE_XY	A661_ParameterStructure_8Bytes
StyleSet	ushort	16	A661_STYLE_SET	A661_Parameter_structure_2Bytes

3.6.4 Focus Navigation Widgets

It is often required to move the cursor focus among widgets residing in different layers, including layers owned by different User Applications. Two ways of supporting this kind of Focus navigation are presented here. One method makes use of a single widget (FocusLink). The other method uses two widgets (FocusIn, FocusOut). For both methods the OEM and CDS provider must take care to manage the configuration of the widgets in the two layers.

Note: The choice of which solution to use is CDS dependent. It is not necessary to implement both solutions.

3.6.4.1 FocusLink

Categories:

Utility

Description:
The FocusLink widget is used to define a NextFocusWidget sequence that crosses between two separate layers. The interactive widgets and the FocusLinks are chained together using the NextFocusedWidget parameters in the normal way.

Figure 3.6.4.1 shows an example of two layers with FocusLinks used to join the focus sequences. To the Pilot the focus would move through the sequence 17, 18, 19, 58, 59. Although the FocusLinks (20 and 57) exist in the chain of NextFocusedWidgets the highlight box does not "focus" on these items.

The NextLayerConnectorRef parameter is used to point to the second layer in the same way as a ConnectorReference is used for the Connector widget.

The value of the NextFocusedWidget is used to point to the next widget in the focus sequence. Note that for the FocusLink the next widget maybe on another UAs layer.

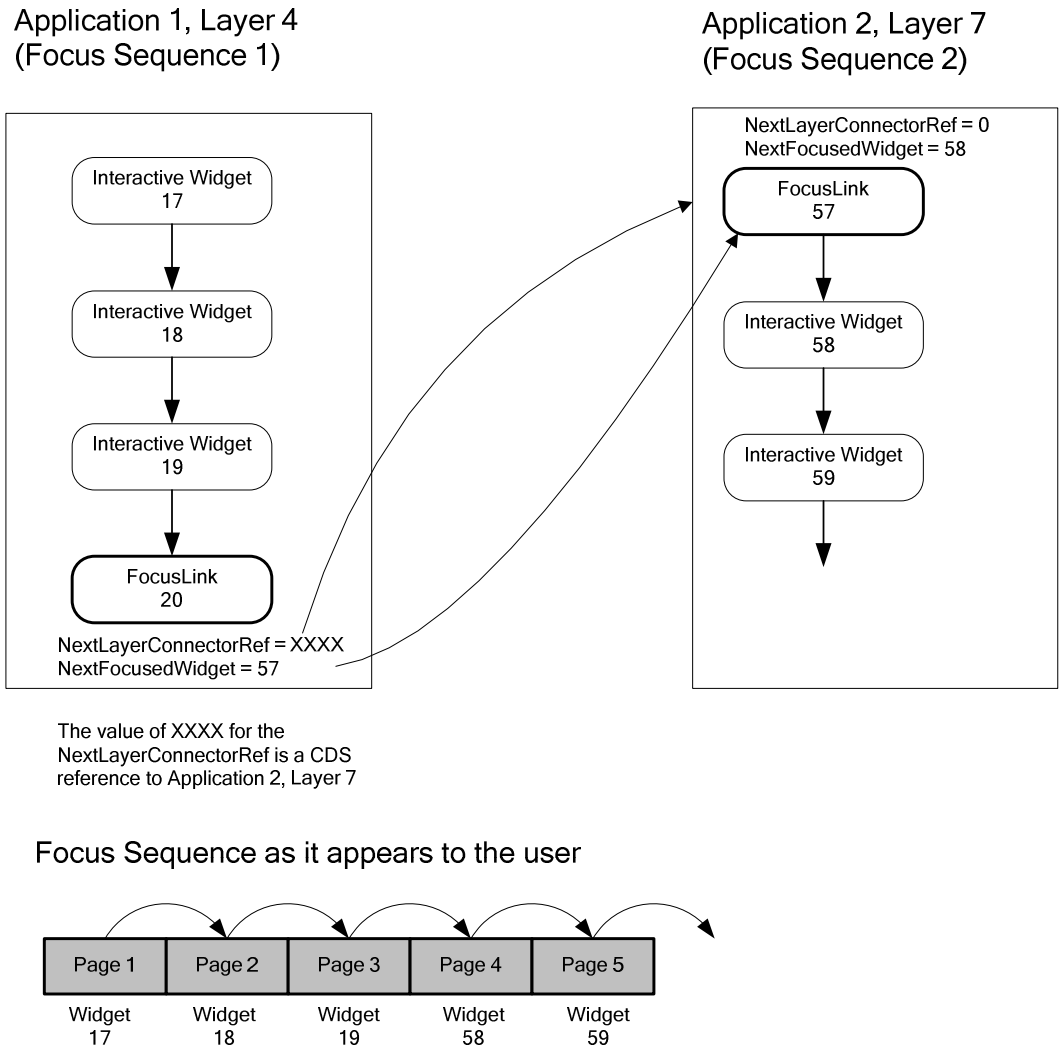


Figure 3.6.4.1 – FocusLink

Restriction:

None

FocusLink Parameters are defined in Table 3.6.4.1-1.

Table 3.6.4.1-1 – FocusLink Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_FOCUS_LINK
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextLayer ConnectorRef	D	CDS reference to the linked layer and/or FocusLink widget associated with this focus sequence. The resolution of the link is a CDS configuration issue.
NextFocusedWidget	DR	Widget ident of next widget in the focus sequence.

FocusLink Creation Structure is defined in Table 3.6.4.1-2.

Table 3.6.4.1-2 – FocusLink Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_LINK
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Unused Pad	N/A	32	
Unused Pad	N/A	16	
NextLayerConnectorRef	ushort	16	
NextFocusedWidget	ushort	16	

Table 3.6.4.1-3 – FocusLink Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED _WIDGET	A661_ParameterStructure_2Bytes

3.6.4.2 FocusIn

Categories:

Utility

Description:
The FocusIn widget is used together with the FocusOut widget to define a NextFocusedWidget sequence that crosses between two separate layers. The interactive widgets and the FocusOut / FocusIn widgets are chained together using the NextFocusWidget parameters in the normal way.

Figure 3.6.4.2 shows an example of two layers with FocusOut / FocusIn used to join the focus sequences. To the Pilot the tabber would move through the sequence 17, 18, 19, 58, 59. Although the FocusOut (20) and FocusIn (57) exist in the chain of NextFocusedWidgets the highlight box does not "focus" on these items (since they does not have any graphical representation).

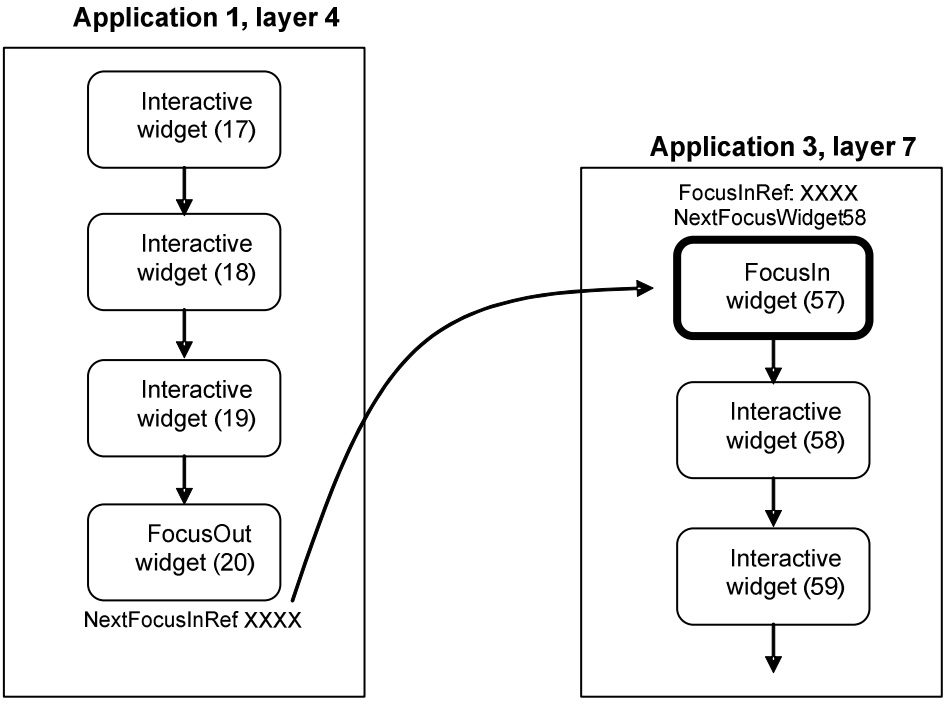


Figure 3.6.4.2 – FocusIn

The connection between the two layers is performed thanks to the FocusInRef : the FocusOut widget identifies the FocusIn widget that will receive the focus through this FocusInRef (NextFocusInRef parameter). The FocusIn widget is associated to this FocusInRef parameter thanks to its FocusInRef parameter.

Each FocusIn widget shall have a unique CDS-wide FocusInRef. The CDS vendor is thus in charge of ensuring that this reference is unique. The behavior of the CDS when to FocusIn widgets have the same FocusInRef is undetermined.

Restriction:

None

FocusIn Parameters are defined in Table 3.6.4.2-1.

Table 3.6.4.2-1 – FocusIn Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_FOCUS_IN
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextFocusedWidget	D	Widget ident of next widget to be focused upon crew member validation

FocusIn Creation Structure is defined in Table 3.6.4.2-2.

Table 3.6.4.2-2 – FocusIn Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_IN
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
NextFocusedWidget	ushort	16	

3.6.4.3 FocusOut

Categories:

Utility

Description:
The FocusOut widget is used together with the FocusIn widget to define a NextFocusedWidget sequence that crosses between two separate layers. The interactive widgets and the FocusOut / FocusIn widgets are chained together using the NextFocusedWidget parameters in the normal way.

Figure 3.6.4.3 shows an example of two layers with FocusOut / FocusIn used to join the focus sequences. To the Pilot the tabber would move through the sequence 17, 18, 19, 58, 59. Although the FocusOut (20) and FocusIn (57) exist in the chain of NextFocusedWidgets the highlight box does not “focus” on these items (since they does not have any graphical representation).

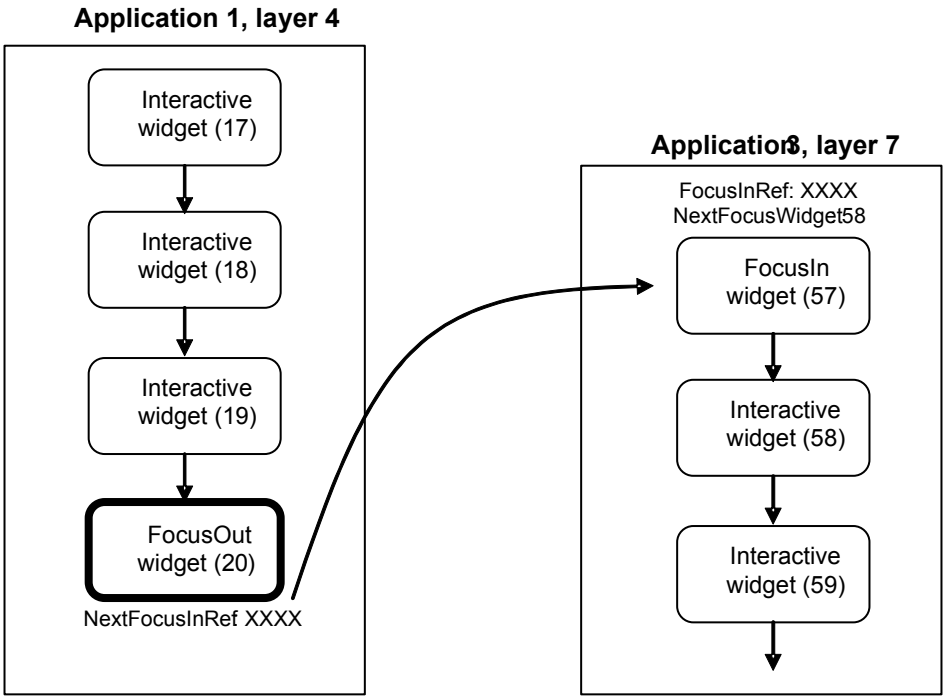


Figure 3.6.4.3 – FocusOut

The connection between the two layers is performed thanks to the FocusInRef the FocusOut widget identifies the FocusIn widget that will receive the focus through this FocusInRef (NextFocusInRef parameter). The FocusIn widget is associated to this FocusInRef parameter thanks to its FocusInRef parameter.

Restriction:

None

FocusOut Parameters are defined in Table 3.6.4.3-1.

Table 3.6.4.3-1 – FocusOut Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_FOCUS_OUT
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextFocusInRef	D	Unique CDS-wide identifier of the FocusIn widget that will receive the focus when this focus reaches the FocusOut widget

FocusOut Creation Structure is defined in Table 3.6.4.3-2.

Table 3.6.4.3-2 – FocusOut Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_OUT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
NextFocusInRef	ushort	16	

3.6.5 SizeToFitContainer

Categories:

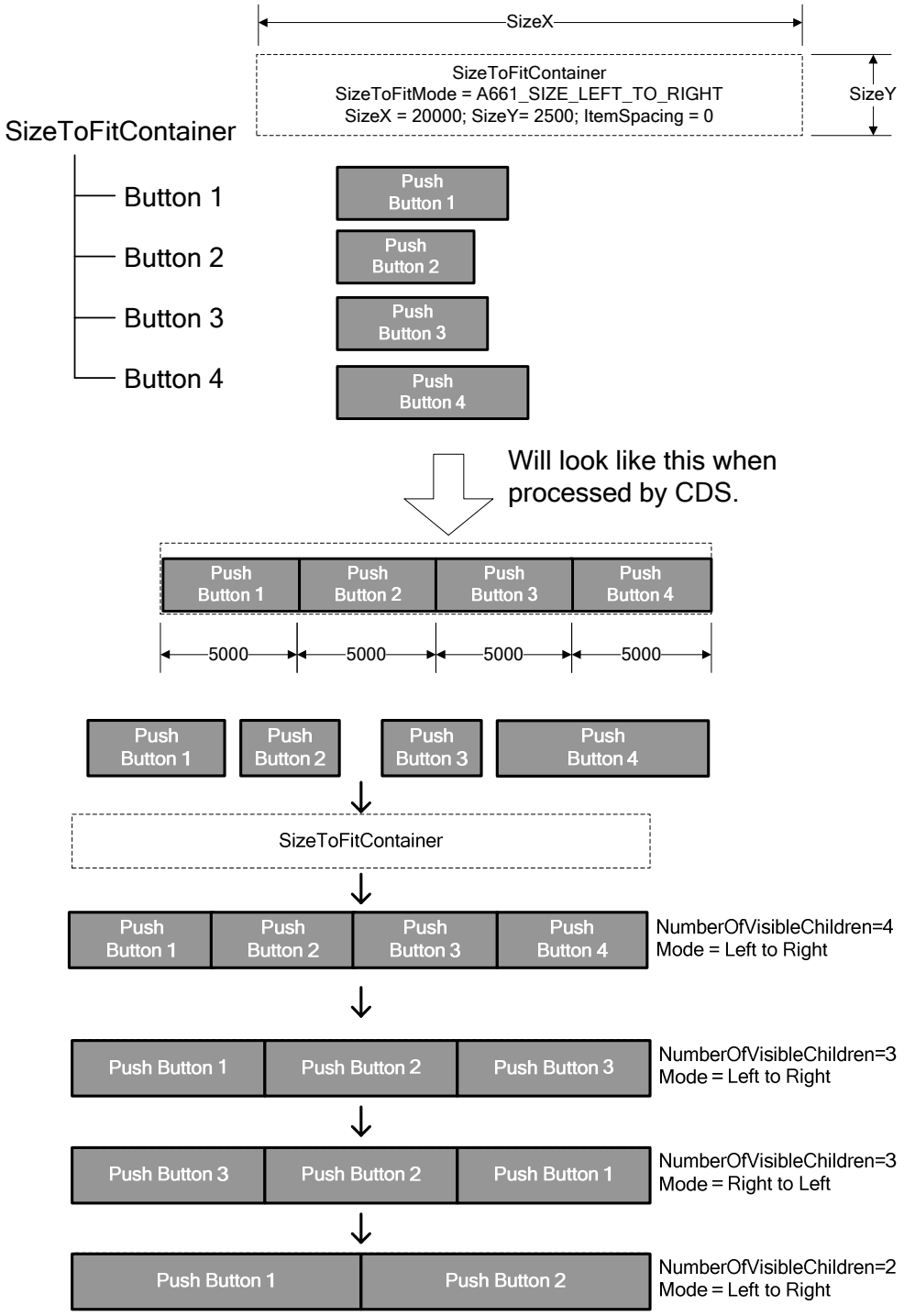
Container

Graphical Representation

Description:
This widget is used to stretch/shrink its children such that they all have the same SizeX or SizeY, and are arranged either horizontally or vertically within the bounds of the SizeToFitContainer widget. The child widgets can also be configured so that they are ordered from top to bottom, left to right, etc.

This is typically useful when constructing menus or other list of controls, while still allowing the UA developer to control the nature of the interactive widgets that are to be sized. The widget function can be equally applied to non interactive widgets such as GpRectangles, etc.

Its purpose is to layout child widgets either in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the SizeToFitContainer.



Note: In this diagram the buttons are aligned vertically for simplicity. This does not indicate an intended function of the widget.

Figure 3.6.5-1 – SizeToFit Example

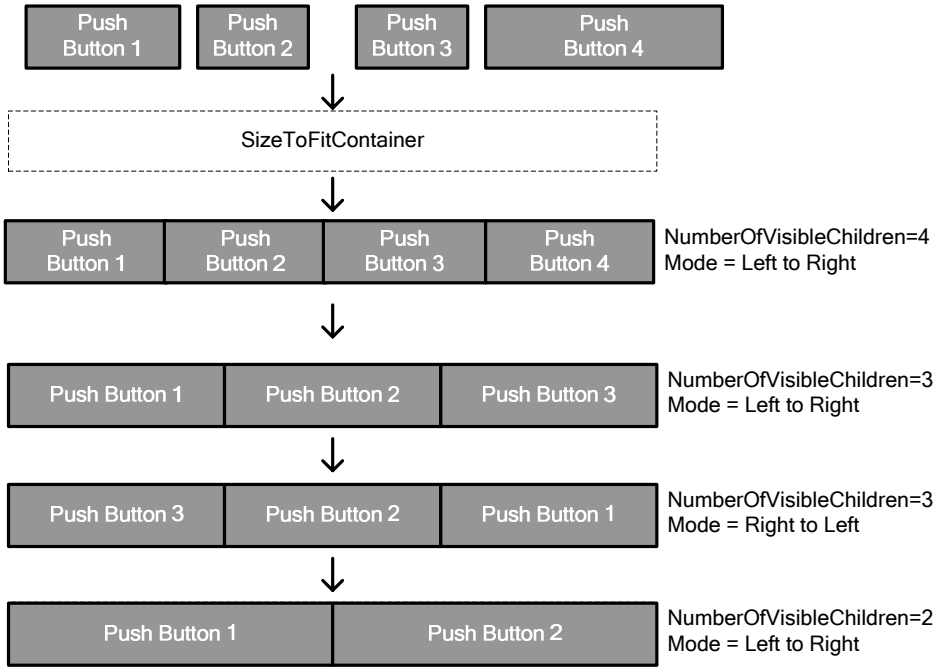


Figure 3.6.5-2 – SizeToFit Example

When the size to fit mode is operating horizontally the widget SizeX and PosX values are computed.

When the size to fit mode is operating vertically the widget SizeY and PosY values are computed.

If a horizontal (Size X) sizing mode is selected, the effect on SizeY of the child widgets is CDS dependent. Similarly if a vertical (SizeY) sizing mode is selected the effect on SizeX of the child widgets is CDS dependent.

The function of the SizeToFitContainer only applies to the first "n" children whose visibility is set to true, where "n" is equal to NumberOfVisibleChildren. If a child widget's visibility is set to false it is not considered when counting the NumberOfVisibleChildren.

The SizeToFitContainer only affects its immediate children. It does this even if the Size and/or Pos attributes of the children are definition time only.

The order of widgets is defined by their order in the Definition File, the order direction is determined by the SizeToFitMode parameter.

The function of the widget can be suspended, and the last computed sizes retained by changing the size to fit mode from any fit mode to "no sizing". When the function is suspended in this way the child widgets retain their last computed size.

Although the SizeToFitContainer is listed as having a graphical representation, typically it will not appear on the display. Its purpose is to lay out child widgets either in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the SizeToFitContainer.

An optional item spacing can be applied between the child widgets. The optional item spacing is only applied in the direction indicated by the sizing mode.

SizeToFitContainers can be one of the children of a SizeToFitContainer.

In the example shown in Figure 3.6.5-3, the first SizeToFitContainer 1 has two visible children, one of which is another SizeToFitContainer 2. SizeToFitContainer 1 allocated half of the horizontal space to Button 1 and half to SizeToFitContainer 2. SizeToFitContainer 2 then sub-allocates is space to its own three children. Each one being sized to be 33.33% of the space allocated to SizeToFitContainer 2.

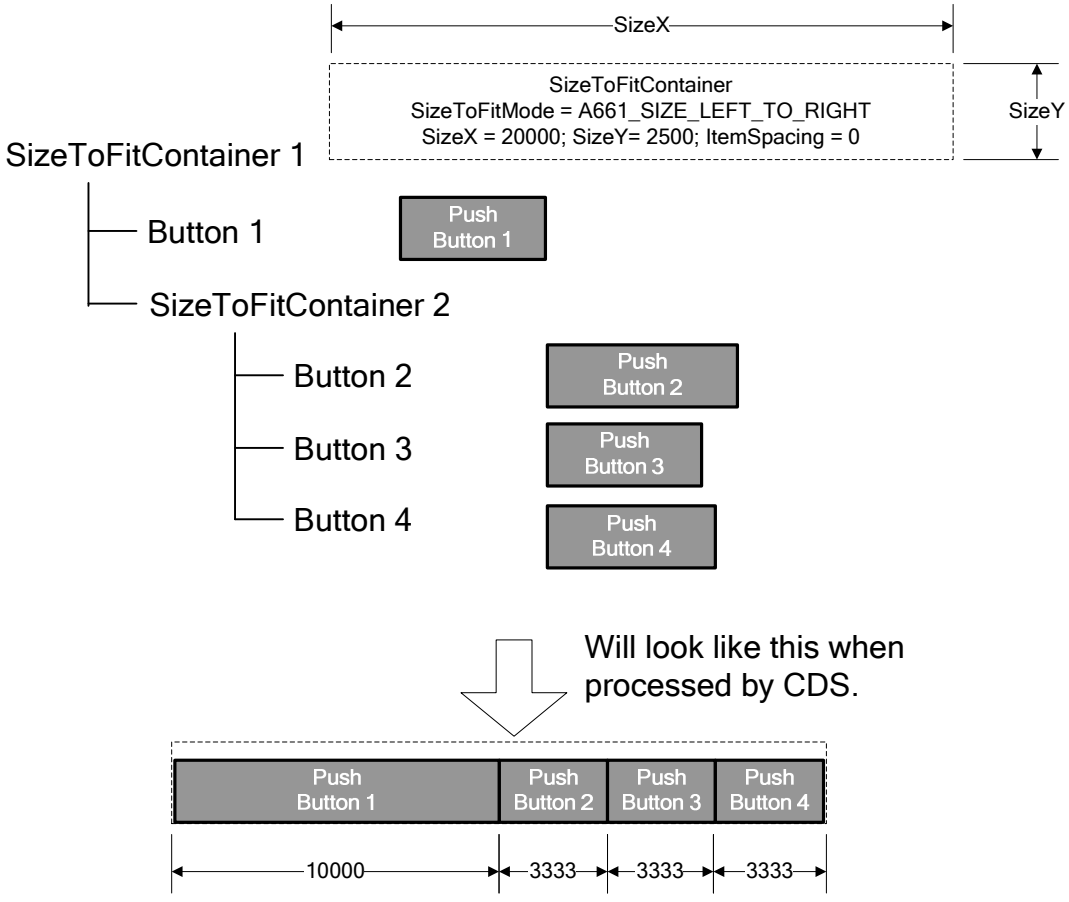


Figure 3.6.5-3 – SizeToFit Example

The effect of nesting SizeToFitContainers or ShuffleToFitContainers is allowed, but the effect is CDS dependent.

Restrictions:

Any widget that has a SizeX / Size Y can be a child of this widget, although its intended use is that is should only contain basic widgets such as Checkbutton, Label, LabelComplex, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ToggleButton and SelectionListButton. If complex widgets are made children of this widget, overly complicated nesting should be avoided.

SizeToFitContainer Parameters are defined in Table 3.6.5-1.

Table 3.6.5-1 – SizeToFitContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SIZE_TO_FIT_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be activated
Visible	DR	Visibility of the widget
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The width of the widget.
SizeY	DR	The height of the widget.
NumberOfVisible Children	DR	Defines the first “n” visible children that to which the sizing function is applied.
SizeToFitMode	DR	Determines how the children in the container are sized to fit. A661_SIZE_TOP_DOWN A661_SIZE_BOTTOM_UP A661_SIZE_LEFT_TO_RIGHT A661_SIZE_RIGHT_TO_LEFT A661_NO_SIZING
ItemSpacing	D	Defines the spacing between the widgets in 1/100th mm, applied in the axis defined by SizeToFitMode, or the last axis if “no sizing” mode is selected.

SizeToFitContainer Creation Structure is defined in Table 3.6.5-2.

Table 3.6.5-2 – SizeToFitContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SIZE_TO_FIT_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
NumberOfVisible Children	ushort	16	
SizeToFitMode	uchar	8	
Unused Pad	N/A	8	
ItemSpacing	ulong	32	

Available SetParameter identifiers and associated data structure are defined in Table 3.6.5-3.

Table 3.6.5-3 – SizeToFitContainer Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_Parameter_structure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_Parameter_structure_1Byte
PosX	long	32	A661_POS_X	A661_Parameter_structure_4Bytes
PosY	long	32	A661_POS_Y	A661_Parameter_structure_4Bytes
PosX PosY	long x 2	32 x 2	A661_POS_XY	A661_Parameter_structure_8Bytes
SizeX	ulong	32	A661_SIZE_X	A661_Parameter_structure_4Bytes
SizeY	ulong	32	A661_SIZE_Y	A661_Parameter_structure_4Bytes
SizeXY	long x 2	32 x 2	A661_SIZE_XY	A661_ParameterStructure_8Bytes
NumberOfVisible Children	ushort	16	A661_NUMBER_OF_ VISIBLE_CHILDREN	A661_Parameter_structure_1Byte
SizeToFitMode	uchar	8	A661_SIZE_TO_FIT_MODE	A661_Parameter_structure_1Byte

3.6.6 ShuffleToFitContainer

Categories:

Container

Graphical Representation

Description:
The ShuffleToFitContainer is used to arrange all of its child widgets in a continuous vertical or horizontal stack. Typically this is used in menus where an item may only be visible some of the time. The remaining items below it are then shuffled up to fill the gap left by the widget(s) that is invisible.

The purpose of the widget is to layout its child widgets either in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the ShuffleToFitContainer.

When the widgets are shuffled horizontally their PosX values are computed. When the widgets are shuffled vertically, their SizeY values are computed.

If horizontal shuffling is selected, the effect on PosY of the child widgets is CDS dependent. Similarly if vertical shuffling is selected the effect on PosX of the child widgets is CDS dependent.

The function of the ShuffleToFitContainer only applies to the first "n" children whose visibility is set to true, where "n" is equal to NumberOfVisibleChildren. If a child widget's visibility is set to false, it is not considered when counting the NumberOfVisibleChildren.

The function of the widget can be suspended, and the last computed positions retained by changing the shuffle to fit mode from any fit mode to "no shuffle". When the function is suspended in this way the child widgets retain their last computed PosX/Y.

Optional item spacing can be applied between the child widgets. The optional item spacing is only applied in the direction indicated by the fit mode.

The ShuffleToFitContainer only affects the PosX/Y of widgets; **it does not affect their Size.**

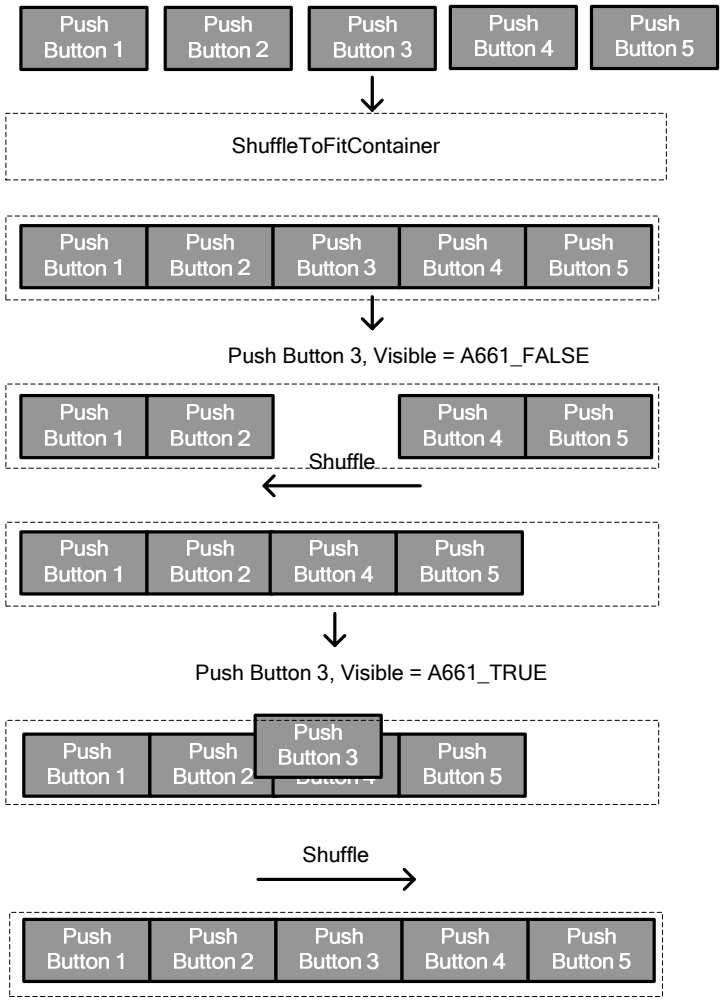


Figure 3.6.6 – ShuffleToFit Example

The effect of nesting SizeToFitContainers or ShuffleToFitContainers is allowed, but the effect is CDS dependent.

Restriction:

Any widget that has a SizeX / Size Y can be a child of this widget, although its intended use is that is should only contain basic widgets such as Checkbutton, Label, LabelComplex, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ToggleButton and SelectionListButton. If complex widgets are made children of this widget, overly complicated nesting should be avoided.

ShuffleToFitContainer Parameters are defined in Table 3.6.6-1.

Table 3.6.6-1 – ShuffleToFitContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SHUFFLE_TO_FIT_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be activated
Visible	DR	Visibility of the widget
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The width of the widget.
SizeY	D	The height of the widget.
NumberOfVisible Children	DR	Defines the first “n” visible children that to which the sizing shuffle function is applied.
ShuffleToFitMode	DR	Determines how the children in the container are re-positioned. A661_SHUFFLE_UP A661_SHUFFLE_DOWN A661_SHUFFLE_LEFT A661_SHUFFLE_RIGHT A661_NO_SHUFFLE
ItemSpacing	D	Defines the spacing between the widgets in 1/100th mm, applied in the axis defined by ShuffleToFitMode.
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.

ShuffleToFitContainer Creation Structure is defined in Table 3.6.6-2.

Table 3.6.6-2 – ShuffleToFitContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SHUFFLE_TO_FIT_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
NumberOfVisibleChildren	ushort	16	
ShuffleToFitMode	uchar	8	A661_SHUFFLE_UP A661_SHUFFLE_DOWN A661_SHUFFLE_LEFT A661_SHUFFLE_RIGHT A661_NO_SHUFFLE
Unused Pad	N/A	8	
ItemSpacing	long	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	

Available SetParameter identifiers and associated data structure are defined in Table 3.6.6-3.

Table 3.6.6-3 – ShuffleToFitContainer Runtime Modifiable Parameters

Name of the parameter to set	Type	Size (bits)	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_Parameter_structure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_Parameter_structure_1Byte
NumberOfVisible Children	uchar	8	A661_NUMBER_OF_ VISIBLE_CHILDREN	A661_Parameter_structure_1Byte
ShuffleToFitMode	uchar	8	A661_SHUFFLE_TO_FIT_ MODE	A661_Parameter_structure_1Byte

4.1 Introduction

This section defines the type, content and format for ARINC 661 data to be exchanged between a User Application (UA) and the Cockpit Display System (CDS). This includes the type, content and format of the data. At definition time, a Definition File (DF) is sent from UAs to the CDS. At run-time, messages are exchanged between UAs and CDS.

4.2 Definition Phase Exchange

4.2.1 Definition File and UALD

A DF contains User Application Layer Definitions (UALDs) from one UA. The UALD describes data shared between the UA and the CDS, as depicted in Figure 4.1.

The DF header and footer are OEM dependent. The data between header and footer are defined in this specification, and are composed of UALDs, using a block structure defined later in this section. Each block (i.e., each UALD) contains the definition of exactly one layer. Support for multiple blocks in a file is OEM dependent.

Each UA displaying data inside the CDS is associated with at least one layer. The UALD describes the hierarchical structure of the UA widgets **residing** in the layer as well as the specific ARINC 661 interface parameters of these widgets. The format of a block is described in Sections 3.0 and 4.5. The hierarchical structure of the widgets is ensured by the use of the “ParentIdent” parameter in each widget definition.

For each type of widget, all parameters must have a CDS default value. If a parameter is not set in the UALD, the CDS default value will be used. This will happen in the case of run-time-only parameters, or in the case where a CDS library definition has been updated to incorporate new parameters, but an older UALD is not updated.

On one side, Figure 4-1 shows widgets “owned” by the UA (primarily, defines their IDs) for display of information on CDS. This is the definition of the static graphical part of the UA interface for one layer.

On the other side, Figure 4-1 shows that the CDS interprets the UALD data to allocate and construct the hierarchical tree of widgets in conjunction with the CDS widget library.

At run-time the CDS and UA exchange ARINC 661 messages to manage the widget parameters, and thus their graphical representation.

4.2.2 Binary Format

The DF describes shared data. Therefore, one main objective of this specification is to standardize the DF data and format shared between the UA and the CDS. The graphical part of the application interface (DF) must be loaded into the CDS. This DF describes only data, and is therefore interpreted by the CDS to be associated with the CDS graphical capacities.

To limit the impact of a UA modification on the CDS, the data format loaded into CDS should be independent from the CDS. Besides, as a growth potential, a data format independent from the CDS should provide the capability to UAs to download their Definition Files (DF) into the CDS.

The format for the DF is a standard non-executable binary format, **allowing uploading and downloading of the DF.**

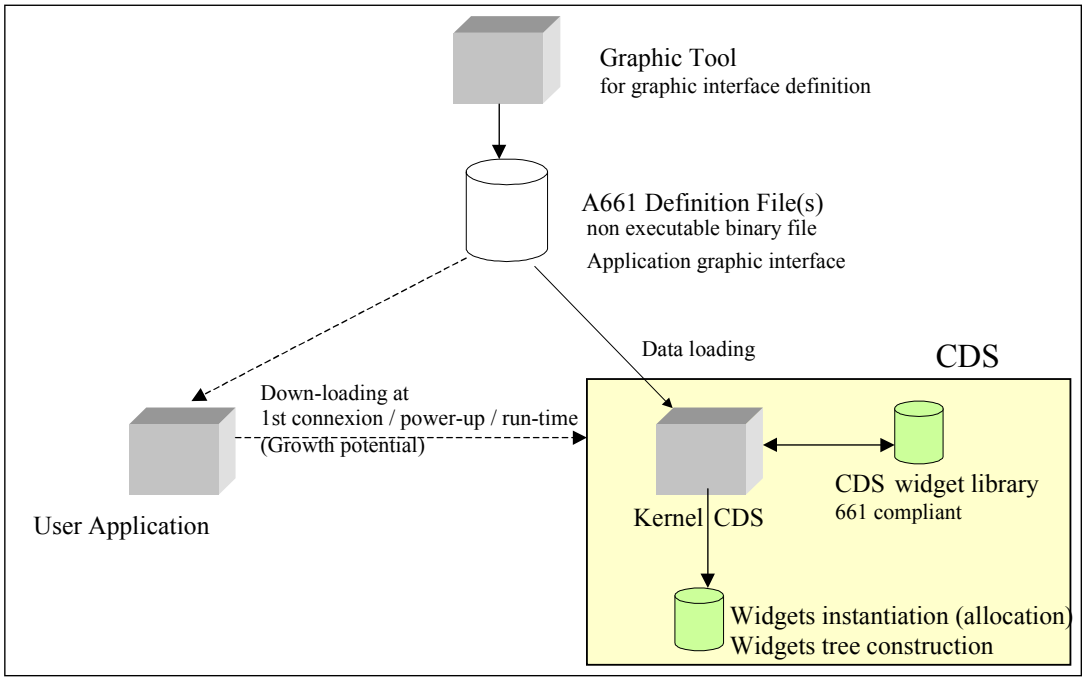


Figure 4.1 – Definition File Integration Process

4.3 Run-time Communication

4.3.1 General Principle

The communication from UA to CDS concerns run-time widget parameters for widgets management. The transmittal of these parameters corresponds to:

- A context change of the application, which could correspond to current periodic parameter transmittal
- A response to a widget event, which is purely asynchronous

The communication from CDS to UA is event driven. The transmittal of these parameters corresponds to:

- Event notification from a widget, which is purely asynchronous

Basically, the communication from UA to CDS is event driven. Therefore, all messages will be sent in an asynchronous way. Asynchronous exchange for UA functional context change should save bandwidth as well as decrease latency-time values. Nevertheless, if synchronous refresh of information is needed for a **particular** UA, the UA could send the information on a periodic basis.

Due to the Client-Server architecture of ARINC 661, a communication channel between a CDS and a User Application is logically a point-to-point

connection, even if the actual network architecture uses multicast or broadcast mechanisms.

COMMENTARY

Concerning widget states, the associated parameters should be managed in an asynchronous way. Periodic transmittal of messages for controlling widget states could raise a “race condition,” described in Appendix A, Glossary.

4.3.2 Issues

The asynchronous approach should mitigate the following design concerns:

1. Loss of a message: From UA to CDS, the display will not be in the state that the UA expects. From CDS to UA, the UA will not react to a crew member interaction. Refer to Section 4.3.3, Assumption on Communication Reliability.
2. Synchronization: A UA needs to be sure that a message will arrive before the next transmitted message. Refer to Section 4.3.3, Assumption on Communication Reliability.
3. Reconfiguration of the rendering unit: The new rendering subsystem may not have needed information available. Refer to Section 2.3.2.4, Layer Activity/Visibility Management.

4.3.3 Assumption on Communication Reliability

This specification is independent of any bus choice. Nevertheless, some assumption should be made on the level of reliability. Therefore, the basic assumption is that the communication is in reliable packet order.

This applies to:

- **The correct reception order of messages, (see Issue 2 above)**
- **The loss of messages, (see Issue 1 above)**

4.3.4 Layer Data Management

Refer to Section 2.3.2.4, Layer Activity/Visibility Management.

4.4 ARINC 661 Commands

4.4.1 Type of Commands

Table 4.4.1-1 describes the definition-time command.

Table 4.4.1-1 – Definition-Time Command

	Command Type	Description
UA ⇓ CDS	A661_CMD_CREATE	Creation of a widget with definition of its parameters. For a version of the Widget library, all the parameters of a used widget should be set. Refer to Section 4.5, ARINC 661 Command Structure. Parameter order inside the Creation buffer is provided in each widget description in Section 3.3, Widget List.

Table 4.4.1-2 describes run-time commands.

Table 4.4.1-2 – Run-time Commands

	Command name	Description
UA ⇓ CDS	A661_CMD_SET_PARAMETER	Set the value of one parameter for one widget. The target of this message is one widget of one UA layer.
	A661_CMD_UA_REQUEST	Request from UA to CDS. A request corresponds to a message exchange between a UA and the CDS, without widget targeted. Request from the UA to the CDS may or may not be accepted by the CDS (refer to section 4.4.3, ARINC 661 Request/Notification.)
CDS ⇓ UA	A661_NOTIFY_WIDGET_EVENT	Notification of an Event from CDS to UA. This message is initiated by a widget on which an interaction has occurred. The nature of the event depends on the widget type and the interaction on this widget. This command also contains the Context Number associated with the current context of the layer.
	A661_NOTIFY_LAYER_EVENT	Notification of a request from CDS to UA. The request is a notification from the CDS. The difference with the previous message type is this notification is initiated at layer level. It corresponds to an event from the Layer managed by the CDS (Refer to Section 4.4.3.2, Request/Notification from CDS to UA).
	A661_NOTIFY_EXCEPTION	Notification of an error from CDS to UA. Refer to Section 4.4.2, Error Notification, for all applicable errors.

4.4.2 Error Notification

This Specification defines the principle of error notification to provide the tools to manage errors in a command. Nevertheless it is outside the scope of this document to define the recovery action on an error notification. The aircraft OEM should specify only error recording in built-in test equipment (BITE), or some recovery mechanism to implement specific errors.

The error notifications which the CDS may send in response to erroneous commands and overload situations are described in Table 4.4.2 in this specification.

This ARINC specification lists valid values that User Applications and the CDS are supposed to use when exchanging run-time messages. The CDS's response to undefined values is up to the OEM/CDS implementation. At the CDS's discretion, error messages may be created and sent to the User Application if for example a value other than "0" and "1" is sent as a Boolean parameter value, or the CDS may choose to fall back to a default value (for

example, interpret every non-zero valid the same as A661_True, or only look at the least significant bit of the 8-bit value). This applies analogous to some other parameter types, such as those using enumerated types. Note that this is not to be understood as an encouragement for a sender of 661 run-time messages to rely on the behavior of a specific CDS implementation when sending data to that CDS, because doing so could lead to an undesirable dependency between a User Application and that CDS.

Any or all of the following notifications are sent at the discretion of the CDS.

Table 4.4.2 – Exception Type

Command/ Request type	A661_ExceptionType	Description
Error notification on command error		
All	A661_ERR_BAD_COMMAND	This exception is sent on any erroneous command. It applies to: Invalid block structure (keyword or size field) Invalid command/request code
Create	A661_ERR_CREATE_ABORTED	This exception is sent on erroneous Create command. It applies to: Invalid Layer ID or Context ID Invalid Widget ID Invalid parameter value Insufficient required parameter data (growth potential for direct downloading the DF from UA to CDS)
SetParameter	A661_ERR_SET_ABORTED	This exception is sent on erroneous SetParameter. It applies to: Invalid Layer ID or Context ID Invalid Widget ID or Parameter ID Invalid parameter value Insufficient parameter data
UARequest	A661_ERR_UA_REQUEST_ABORTED	This exception is sent on erroneous UA Request. It applies to: Invalid Layer ID or Context ID Invalid Request key value Invalid Widget ID Insufficient required parameter data
Error notification on CDS Resource overload		
	A661_ERR_MEMORY_OVERLOAD	Notification of memory overloading by allotting UA widgets. (Definition time). (Growth potential for direct downloading the DF from UA to CDS)
	A661_ERR_PROCESS_OVERLOAD	Inability to complete processing of desired image.
	A661_ERR_RENDERING_OVERLOAD	Inability to complete rendering of desired image.

Because the level of CDS is the higher application, there is no need for exceptions on commands from CDS to UA.

4.4.3 ARINC 661 Request/Notification

Communication described in this specification is based on the widget management. Requests apply to messages exchanged between UA and the CDS without a particular widget being targeted. These messages provide a means for the UA to change HMI mechanisms under CDS responsibility, such as Focus position or layer activity. In the other direction, these messages provide the CDS a means to indicate the current state to the UA.

4.4.3.1 Request from UA to CDS

Requests from the UA to the CDS, described in Table 4.4.3.1, may or may not be accepted by the CDS. These requests should be sent to the CDS through A661_CMD_UA_REQUEST command.

Table 4.4.3.1 – Request from UA to CDS

Request Type	Description
A661_REQ_LAYER_ACTIVE	Provide a means for a UA application to request to the CDS the activation of its layer. The CDS may or may not accept the request according to the current possible configuration. When a layer is active, the CDS should update widget parameters of this layer (refer to Section 2.3.2.4 – Layer Activity/Visibility Management).
A661_REQ_LAYER_INACTIVE	Provide a means for a UA to request the CDS to deactivate its layer.
A661_REQ_FOCUS_ON_WIDGET	Move focus on a defined widget of one UA layer.
A661_REQ_LAYER_VISIBLE	Turn on the visibility of one layer. This request follows the CDS notification of the layer activity.
A661_REQ_CURSOR_ON_WIDGET	**Move the cursor to a defined widget of one UA layer.**

4.4.3.2 Request/Notification from CDS to UA

Request/Notifications, described in Table 4.4.3.2, should be sent from CDS to the UA through the A661_NOTIFY_LAYER_EVENT command. The UA should to take into account the notification.

Table 4.4.3.2 – Request/Notification from CDS to UA

Request/Notification Type	Description
A661_NOTE_REINIT_LAYER_DATA	CDS request to the UA for Layer data initialization. The response of the UA should be a block of SetParameter commands.
A661_NOTE_LAYER_IS_ACTIVE	CDS notification to the UA that its layer becomes active. This implies that the UA will reinitialize the layer data.
A661_NOTE_LAYER_IS_INACTIVE	Notification of layer deactivation.

4.5 ARINC 661 Command Structure

4.5.1 Notation

Notations used in ARINC 661 Command Structures:

Command	Description
<A>	element of type A
“<A> | <B>”	means <A> or <B>
“{<A>}”	means a set of 0 or more of <A>
“{<A>}+”	means a set of 1 or more of <A>
“(…)”	are used for external references or comments
[<A>]	means zero or one of <A>

4.5.2 Block Structure

The UA and CDS exchange information through blocks of commands. A block represents a set of data to be processed as coherent information.

The structures detailed in Tables 4.5.3.1-2 and 4.5.4.1-1 are built so that a single datum (excluding arrays) is never encoded across two 32-bit words. This simple rule is emphasized for better understanding of the structure details.

4.5.3 Definition Time Exchanged Structure

4.5.3.1 UADF Loading Structure

The DF is recommended to be loadable into CDS according to the ARINC 615A standard. The format of the loadable software is described in the ARINC 665 standard. In this case the DF is defined as a data file.

According to A665, the data file will be a binary file with the extension XXX.LUP. The 665 header file has the extension: XXX.LUH. The header file defines which data file(s) are to be loaded.

It is recommended that only one data file be attached to the A665 header of a DF. In this case the “Number of Data Files” (in A665 header) would be 1.

It is recommended that no support file be attached to the A665 header of a data file. The “Number of Support Files” (in A665 header) should be 0.

4.5.3.2 Definition File (DF) Structure

A Definition File (DF) may hold several User Application Layer Definitions (UALD) from one UA, zero or more symbol definitions **and zero or more picture definitions**. The loadable entity is the DF. Table 4.5.3.2-1 describes the structure of a DF.

Table 4.5.3.2-1 – Definition File Structure

A661_Definition_File	Description
DF_File_Header	Header of the file
{A661_Symbol_Block_Structure_DT}	Defines ARINC 661 Symbols
{A661_Picture_Block_Structure_DT}+	**Defines ARINC 661 Pictures (bitmaps) (see Section 7 for details)**
{ **A661_Block_Structure_DT** }+	Core of the file
DF_File_Footer	Footer of the file

The DF has a header with a OEM specific part:

Table 4.5.3.2-2 – Definition File Header

DF_File_Header	Type	Size (bits)	Description
A661_DF_MAGIC_NUMBER	ushort	16	Identifier for A661 Definition File
LibraryVersion	uchar	8	Identifier of the Library version compatible with the User Application Definition File. This value is implementation dependent.
SuppVersion	uchar	8	Identifier of A661 Supplement version on which the User Application Definition File is based. A661 & A661-1 value = “00” A661-2 value = “02” A661-3 value = “03”
AppliIdent	ushort	16	Identifier of the User Application.
Size of the OEM free data	ushort	16	**Size of the OEM_Free_Data in bytes**
OEM_Free_Data	N/A	{32}+	

Note: The DF has a footer.

Table 4.5.3.2-3 – Definition File Footer

DF_File_Footer	Type	Size (bits)	Description
A661_DF_FOOTER	uchar	8	Keyword to indicate the end of the Definition File
UnusedPad	N/A	24	0

Block Structure exchanged between UA and CDS at definition time is defined in Tables 4.5.3.2-4 and 4.5.3.2-5.

Table 4.5.3.2-4 – Block Structure Exchanged Between UA and CDS at Definition Time

A661_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_LAYER_BLOCK	uchar	8	Start keyword opening a block of information
LayerIdent	uchar	8	Relative Identifier of the layer for the User Application
Context Number	ushort	16	ContextNumber value attached to the layer. This value is attached by the CDS to any block of message sent before communication is established with UA.
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Definition_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_LAYER_BLOCK	uchar	8	Keyword ending a block of information.
UnusedPad	N/A	24	0

Table 4.5.3.2-5 – Symbol Block Structure Exchanged Between UA and CDS at Definition Time

A661_Symbol_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_SYMBOL_BLOCK	uchar	8	Start keyword opening a symbol definition.
UnusedPad	N/A	24	
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Symbol_Definition_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_SYMBOL_BLOCK	uchar	8	Keyword ending a block of symbol information.
UnusedPad	N/A	24	0

4.5.3.2.1 Definition Time Block Commands

One or more A661_Definition_Command can be included in a block.
Table 4.5.3.2.1 lists the commands defined in the protocol.

Table 4.5.3.2.1 – Definition Time Block Commands

A661_Definition_Command	Type	Size (bits)	Description
A661_Create_Structure	N/A	{32}+	Command applicable at definition time.

4.5.3.3 Command Structure

Table 4.5.3.3 defines the command structure.

Table 4.5.3.3 – Command Structure

A661_Create_Structure	Type	Size (bits)	Description
A661_CMD_CREATE	ushort	16	Start keyword for opening the create structure
CommandSize	ushort	16	Size of the command, in bytes.
CreateParameterBuffer	N/A	{32}+	Refer to the widget library section for the description of all the creation parameter buffers.

4.5.3.4 Constraints Inside a UALD Block

The User Application Layer Definition is composed of LayerBlocks, between A661_BEGIN_LAYER_BLOCK and A661_END_LAYER_BLOCK codes, as follows:

- A UALD LayerBlock should contain only A661_CMD_CREATE commands
- The LayerBlock should contain the complete description of the widgets inside one layer. It implies that a UALD can not be described over several LayerBlock
- Inside a LayerBlock, a widget should be created (with A661_CMD_CREATE command, refer to Section 4.4) after its parent (as defined by ParentIdent parameter, refer to Section 3.1.3.1)
- For reference from a widget to another widget without ParentIdent, there is no restriction on the definition order. For instance, the ActiveTabbedPanelID parameter of the TabbedPanelGroup will be set before the TabbedPanel is actually referenced in the block. However, the consistency of the DF should be checked

4.5.3.5 Definition Time Symbol Block Commands

One or more A661_Symbol Definition_Commands can be included in a block. Table 4.5.3.5 lists the commands defined in the protocol.

Table 4.5.3.5 – Definition Time Symbol Block Commands

A661_Symbol_Definition_Command	Type	Size (bits)	Description
A661_Create_Symbol_Structure	N/A	{32}+	Command applicable at definition time.

4.5.3.6 Symbol Command Structure

Table 4.5.3.6 defines the symbol command structure.

Table 4.5.3.6 – Symbol Command Structure

A661_Create_Symbol _Structure	Type	Size (bits)	Description
A661_CMD_CREATE_SYMBOL	ushort	16	Start keyword for opening the create structure
CommandSize	ushort	16	Size of the command, in bytes.
SymbolId	ushort	16	Symbol Id Value
UnusedPad	N/A	16	
CreateParameterBuffer	N/A	{32}+	Refer to the symbol definition section for the description of all the creation parameter buffers.

4.5.3.7 Constraints Inside a Symbol Definition Block

A symbol can be split into up to **four** different representations:

- The standard representation
- The focus representation
- The highlight representation
- **The sensitive area representation**

Refer to Section 5 for a description of the symbol definition commands, as well as any other restrictions on symbol definitions.

4.5.4 Run-Time Exchange Structure

4.5.4.1 Run-Time Block Commands

One or more A661_Run-Time_Command can be included in a block. Run-time structures do not have any CDS- or OEM-specific header/footer. They may have bus-specific or network-specific packaging at the transport level.

Table 4.5.4.1-1 – Block Structure Exchanged Between UA and CDS at Run Time

A661_Block_Structure_RT	Type	Size (bits)	Description
A661_BEGIN_BLOCK	uchar	8	Start keyword opening a block of information.
LayerIdent	uchar	8	Relative Identifier of the layer for the User Application
Context Number	ushort	16	ContextNumber value attached to one layer. UA->CDS : Value to be returned by CDS with subsequent blocks. CDS->UA : Value attached by UA on last received block.
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Run-Time_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_BLOCK	uchar	8	Keyword ending a block of information.
UnusedPad	N/A	24	0

Table 4.5.4.1-2 lists commands defined in the protocol:

Table 4.5.4.1-2 – Run-Time Block Commands

A661_Run-Time_Command	Description
A661_Set_Parameter_Structure A661_Widget_Event_Structure A661_UA_Request_Structure A661_CDS_Notification_Structure A661_Error_Notification_Structure	Commands applicable at run-time.

The following sections define run-time commands and event notifications.

4.5.4.2 Command Structure – Run-Time Commands

Table 4.5.4.2-1 – Set_Parameter_Structure

A661_Set_Parameter_Structure	Type	Size (bits)	Description
A661_CMD_SET_PARAMETER	ushort	16	Start keyword for opening the set parameter structure.
CommandSize	ushort	16	Size of the command, in bytes.
WidgetIdent	ushort	16	Identifier of the widget
UnusedPad	N/A	16	0
{A661_ParameterStructure}+	N/A	{32}+	Set of parameters with the associated values. Refer to Section 4.5.4.5.

Table 4.5.4.2-2 – UA_Request_Structure

A661_UA_Request_Structure	Type	Size (bits)	Description
A661_CMD_UA_REQUEST	ushort	16	Start keyword for opening the UA request structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_Request_Structure	N/A	{32}+	Type of request from UA to CDS. Refer to Section 4.5.4.3.

Table 4.5.4.2-3 – Widget_Event_Structure

A661_Widget_Event_Structure	Type	Size (bits)	Description
A661_NOTIFY_WIDGET_EVENT	ushort	16	Start keyword for opening the widget event structure.
CommandSize	ushort	16	Size of the command, in bytes.
WidgetIdent	ushort	16	Identifier of the widget
EventOrigin	ushort	16	Identifier of the input device which has been used to initiate the event: (Enumeration to be defined by OEM)
EventStructure	N/A	{32}+	Refer to the widget library for the structure of each widget associated events.

Table 4.5.4.2-4 – CDS_Notification_Structure

A661_CDS_Notification_Structure	Type	Size (bits)	Description
A661_NOTIFY_LAYER_EVENT	ushort	16	Start keyword for opening the CDS notification structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_Layer_Notification_Structure	N/A	{32}+	Type of notification from UA to CDS. Refer to Section 4.5.4.4.

Table 4.5.4.2-5 – Error_Notification_Structure

A661_Error_Notification_Structure	Type	Size (bits)	Description
A661_NOTIFY_EXCEPTION	ushort	16	Start keyword for opening the error notification structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_ExceptionType	ushort	16	Type of error to be notified. Refer to Table 4.4-2.
UnusedPad	N/A	16	0
OEM_free_data	N/A	{32}+	OEM may or may not add free data according to specified mechanism for recovering the error.

4.5.4.3 Request Structure

Table 4.5.4.3-1 – Request_Structure

A661_Request_Structure	Description
A661_Layer_Active_Struct A661_Layer_Inactive_Struct A661_Focus_On_Widget_Struct A661_Layer_Visible_Struct	Type of request

Table 4.5.4.3-2 – Layer_Active_Structure

A661_Layer_Active_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_ACTIVE	ushort	16	Start keyword for opening the layer active structure
UnusedPad	N/A	16	0

Table 4.5.4.3-3 – Layer_Inactive_Structure

A661_Layer_Inactive_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_INACTIVE	ushort	16	Start keyword for opening the layer inactive structure
UnusedPad	N/A	16	0

Table 4.5.4.3-4 – Focus_On_Widget_Structure

A661_Focus_On_Widget_Structure	Type	Size (bits)	Description
A661_REQ_FOCUS_ON_WIDGET	ushort	16	Start keyword for opening the focus on widget structure.
WidgetIdent	ushort	16	Identifier of the widget on which the CDS should move the focus.

Table 4.5.4.3-5 – Layer_Visible_Structure

A661_Layer_Visible_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_VISIBLE	ushort	16	Start keyword for turning on the visibility of one layer.
UnusedPad	N/A	16	0

Table 4.5.4.3-6 – Cursor_On_Widget_Structure

A661_Cursor_On_Widget_Structure	**Type**	**Size (bits)**	**Description**
A661_REQ_CURSOR_ON_WIDGET	**ushort**	**16**	**Start keyword for opening the cursor on widget structure.**
WidgetIdent	**ushort**	**16**	**Identifier of the widget on which the CDS should move the cursor.**
OEM Data Field	**N/A**	**32**	**This could be used for OEM-dependent behavior, e.g. to define which cursor to move**

4.5.4.4 Notification Structure

Table 4.5.4.4-1 – Layer_Notification_Structure

A661_Layer_Notification_Structure	Description
A661_Layer_Is_Active_Struct | A661_Layer_Is_Inactive_Struct | A661_Reinitialize_Layer_Data_Struct	Type of notification

Table 4.5.4.4-2 – Layer_Is_Active Structure

A661_Layer_Is_Active_Structure	Type	Size (bits)	Description
A661_NOTE_LAYER_IS_ACTIVE	ushort	16	Start keyword for opening the layer active structure.
UnusedPad	N/A	16	0

Table 4.5.4.4-3 – Layer_Is_Inactive_Structure

A661_Layer_Is_Inactive_Structure	Type	Size (bits)	Description
A661_NOTE_LAYER_IS_INACTIVE	ushort	16	Start keyword for opening the layer inactive structure.
UnusedPad	N/A	16	0

Table 4.5.4.4-4 – Reinitialize_Layer_Data Structure

A661_Reinitialize_Layer_Data_ Structure	Type	Size (bits)	Description
A661_NOTE_REINIT_LAYER_DATA	ushort	16	Start keyword for opening the reinitialize layer data structure.
UnusedPad	N/A	16	0

4.5.4.5 ARINC 661 Parameter Structure

Section 3.0, Widget Library, provides a description for each widget that includes a table of parameters modifiable at run-time. These tables contain the name of a A661_ParameterStructure, which should be applied to set this parameter. This section provides details of these structures.

For a few specific parameters, the A661_ParameterStructure is defined in the Widget Library. For completeness, all the structures are listed here, with references as necessary.

4.5.4.5.1 A661_ParameterStructure_1Byte

Table 4.5.4.5.1-1 – ParameterStructure_1Byte

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
ParameterValueBuffer	uchar	8	Values associated with the parameter type
UnusedPad	N/A	8	0

4.5.4.5.2 A661_ParameterStructure_2Bytes

Table 4.5.4.5.2-1 – ParameterStructure_2Bytes

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
ParameterValueBuffer	ushort / short	16	Values associated with the parameter type

4.5.4.5.3 A661_ParameterStructure_4Bytes

Table 4.5.4.5.3 -1 – ParameterStructure_4Bytes

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
ParameterValueBuffer	long / ulong / float / fr()	32	Values associated with the parameter type

4.5.4.5.4 A661_ParameterStructure_String

Table 4.5.4.5.4-1 – ParameterStructure_String

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
String size	ushort	16	Size of the string, in bytes, including terminating NULL.
ParameterValue	string	{32}+	List of char, terminated by NULL. Padded by zero, one, two, or three NULL character(s) to be 32 bit aligned

4.5.4.5.5 A661_ParameterStructure_StringArray

Table 4.5.4.5.5-1 – ParameterStructure_StringArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_STRING_ARRAY
Number of Strings	ushort	16	Integer Number of Strings modified by the command
{stringarray_cellstructure}+	N/A	{32}+	

4.5.4.5.6 A661_StringArray_CellStructure

Table 4.5.4.5.6-1 – StringArray_CellStructure

A661_ParameterStructure	Type	Size (bits)	Description
StringIndex	ushort	16	Index of the string
string size	ushort	16	Integer Size of the string, in bytes, including terminating NULL.
String	string	{32}+	List of char. Ended by one, two, three or four NULL character(s) to be 32 bits aligned

4.5.4.5.7 A661_ParameterStructure_EnableArray

Table 4.5.4.5.7-1 – A661_ParameterStructure_EnableArray

A661_ParameterStructure	Type	Size (bits)	Description
Parameter_ident	ushort	16	A661_ENABLE_ARRAY
EntryIndex	uchar	8	
Enable	uchar	8	A661_FALSE A661_TRUE

4.5.4.5.8 A661_ParameterStructure_8Bytes

Table 4.5.4.5.8-1 – ParameterStructure_8Bytes

A661_ParameterStructure_8 Bytes	Type	Size (bits)	Description
Parameter_ident	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
ParameterValue1	N/A	32	
ParameterValue2	N/A	32	

4.5.4.5.9 A661_ParameterStructure_BufferOfItems

For usage with MapHorz_ItemList widget**, see** description in Section 3.3.22.2**.2**.
For usage with MapVert_ItemList widget, see description in Section 3.4.5.2.2.

4.5.4.5.10 A661_ParameterStructure_Buffer

A661_ParameterStructure_Buffer is defined in Table 4.5.4.5.10.

Refer also to BufferFormat widget description in Section 3.3.4 and the BufferFormat data alignment and padding info in Section 3.3.4.1.

Table 4.5.4.5.10 – A661_ParameterStructure_Buffer

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_BUFFER_OF_PARAM
Size	ushort	16	Size of BufferFormatData in bytes.
{BufferFormatData}+	N/A	{32}+	

4.5.4.5.11 A661_ParameterStructure_EntryPopUpArray

Refer to PopUpMenu widget description in Section 3.3.31.1.

4.6 ARINC 661 Keyword Values

The following tables define numeric values associated with the ARINC 661 keywords:

Table 4.6-1 – Constant – Definition File (16 bits)

ARINC 661 Constant – Definition File (16 bits)	
A661_DF_MAGIC_NUMBER	0xA661

Table 4.6-2 – Block Codes (8 bits)

ARINC 661 Block Codes (8 bits)	
A661_BEGIN_LAYER_BLOCK	0xA0
A661_BEGIN_BLOCK	0xB0
A661_BEGIN_PICTURE_BLOCK	**0xBB**
A661_END_PICTURE_BLOCK	**0xBE**
A661_BEGIN_PICTURE	**0xBC**
A661_END_PICTURE	**0xBD**
A661_END_LAYER_BLOCK	0xC0
A661_END_BLOCK	0xD0
A661_DF_FOOTER	0xE0
A661_BEGIN_SYMBOL_BLOCK	0xF0
A661_END_SYMBOL_BLOCK	0xF8

Table 4.6-3a – Commands

ARINC 661 Commands	
A661_CMD_CREATE	0xCA01
A661_CMD_SET_PARAMETER	0xCA02
A661_CMD_UA_REQUEST	0xCA03
A661_CMD_CREATE_SYMBOL	0xCA04

Table 4.6-3b – Commands

ARINC 661 Symbol Definition Commands (16 bits)	
A661_SYMBOL_DEFN_FOCUS	0x9000
A661_SYMBOL_DEFN_HIGHLIGHT	0x9010
A661_SYMBOL_DEFN _SET_COLOR	0x9020
A661_SYMBOL_DEFN _SET_LINE_STYLE	0x9030
A661_SYMBOL_DEFN _SET_FONT	0x9040
A661_SYMBOL_DEFN _SET_HALO	0x9050
A661_SYMBOL_DEFN_LEGEND_ANCHOR	0x9060
A661_SYMBOL_DEFN _ARC_ELLIPSE	0x9070
A661_SYMBOL_DEFN _ARC_CIRCLE	0x9080
A661_SYMBOL_DEFN _CROWN	0x9090
A661_SYMBOL_DEFN _LINE	0x90A0
A661_SYMBOL_DEFN_LINE_POLAR	0x90B0
A661_SYMBOL_DEFN _POLYLINE	0x90C0
A661_SYMBOL_DEFN _RECTANGLE	0x90D0
A661_SYMBOL_DEFN _TRIANGLE	0x90E0
A661_SYMBOL_DEFN _TRIANGLE_FAN	0x90F0
A661_SYMBOL_DEFN _TRIANGLE_STRIP	0x9100
A661_SYMBOL_DEFN _TEXT	0x9110
A661_SYMBOL_DEFN_RECTANGULAR	**0x9120**
A661_SYMBOL_DEFN_CIRCULAR	**0x9130**
A661_SYMBOL_DEFN_SIZE	**0x9140**

Table 4.6-4 – Notifications

ARINC 661 Notifications	
A661_NOTIFY_EXCEPTION	0xCC03
A661_NOTIFY_LAYER_EVENT	0xCC02
A661_NOTIFY_WIDGET_EVENT	0xCC01

Table 4.6-5 – Requests/Notifications

ARINC 661 Requests/Notifications	
A661_REQ_LAYER_ACTIVE	0xDA01
A661_REQ_LAYER_INACTIVE	0xDA02
A661_REQ_FOCUS_ON_WIDGET	0xDA03
A661_REQ_LAYER_VISIBLE	0xDA04
A661_REQ_CURSOR_ON_WIDGET	**0xDA05**
A661_NOTE_REINIT_LAYER_DATA	0xDC01
A661_NOTE_LAYER_IS_ACTIVE	0xDC02
A661_NOTE_LAYER_IS_INACTIVE	0xDC03

Table 4.6-6 – ExceptionType

A661_ExceptionType	
A661_ERR_BAD_COMMAND	0xF001
A661_ERR_CREATE_ABORTED	0xF002
A661_ERR_SET_ABORTED	0xF003
A661_ERR_UA_REQUEST_ABORTED	0xF004
A661_ERR_MEMORY_OVERLOAD	0xF005
A661_ERR_PROCESS_OVERLOAD	0xF006
A661_ERR_RENDERING_OVERLOAD	0xF007

Table 4.6-7 – Widgets (16 bits)

ARINC 661 Widgets (16 bits)	
A661_ACTIVE_AREA	0xA010
A661_BASIC_CONTAINER	0xA020
A661_BLINKING_CONTAINER	0xA030
A661_BUFFER_FORMAT	0xA040
A661_CHECK_BUTTON	0xA050
Reserved	0xA060
A661_COMBO_BOX	0xA070
A661_CONNECTOR	0xA080
A661_CURSOR_POS_OVERLAY	0xA090
A661_EDIT_BOX_MASKED	0xA0A0
A661_EDIT_BOX_NUMERIC	0xA0C0
A661_EDIT_BOX_NUMERIC_BCD	0xA0C2
A661_EDIT_BOX_TEXT	0xA0D0
A661_GP_ARC_CIRCLE	0xA0F0
A661_GP_ARC_ELLIPSE	0xA100
A661_GP_CROWN	0xA110
A661_GP_LINE	0xA120
A661_GP_LINE_POLAR	0xA130
A661_GP_RECTANGLE	0xA140
A661_GP_TRIANGLE	0xA150
A661_LABEL	0xA160
A661_LABEL_COMPLEX	0xA170
A661_MAP_HORZ_ITEMLIST	0xA180
A661_MAP_LEGACY	0xA190
A661_MAP_HORZ_SOURCE	0xA1A0
A661_MAP_HORZ	0xA1B0
A661_MASK_CONTAINER	0xA1C0
Reserved	0xA2E0
Reserved	0xA1E0

ARINC 661 Widgets (16 bits)	
A661_PANEL	0xA1F0
A661_PICTURE	0xA200
A661_PICTURE_PUSH_BUTTON	0xA240
A661_PICTURE_TOGGLE_BUTTON	0xA250
Reserved	0xA260
A661_POP_UP_MENU	0xA270
A661_POP_UP_MENU_BUTTON	0xA280
A661_POP_UP_PANEL	0xA290
A661_PUSH_BUTTON	0xA2A0
A661_RADIO_BOX	0xA2B0
Reserved	0xA2C0
A661_ROTATION_CONTAINER	0xA2D0
A661_SCROLL_LIST	0xA2F0
A661_SCROLL_PANEL	0xA300
A661_SYMBOL	0xA310
A661_TABBED_PANEL	0xA320
A661_TABBED_PANEL_GROUP	0xA330
A661_TOGGLE_BUTTON	0xA340
Reserved	0xA350
A661_TRANSLATION_CONTAINER	0xA360
A66l_MAP_GRID	0xA178
A661_EXTERNAL SOURCE	0xA188
A661_MAP_VERT	0xA1B2
A661_MAP_VERT_SOURCE	0xA1B4
A661_MAP_VERT_ITEMLIST	0xA1B3
A661_EDIT_BOX_MULTI_LINE	0xA0B0
A661_COMBO_BOX_EDIT	0xA0E0
A661_MENU_BAR	0xA1D0
A661_MUTUALLY_EXCLUSIVE_CONTAINER	0xA400
A661_PICTURE_ANIMATED	0xA210
A661_PROXY_BUTTON	0xA420
A661_SELECTION_LIST_BUTTON	0xA370
A661_SLIDER	0xA440
A661_SYMBOL_ANIMATED	0xA450
A661_WATCHDOG_CONTAINER	0xA460
A661_CURSOR_REF	**0xA470**
A661_CURSOR_OVER	**0xA480**
A661_FOCUS_LINK	**0xA490**
A661_FOCUS_IN	**0xA498**
A661_FOCUS_OUT	**0xA499**
A661_SIZE_TO_FIT_CONTAINER	**0xA4A0**
A661_SHUFFLE_TO_FIT_CONTAINER	**0xA4B0**
Reserved for OEM extensions	0xA800 to 0xAFFF

Table 4.6-8 – Parameter Types

ARINC 661 Parameter Types (16 bits)	
A661_AC_LAT	0xB010
A661_AC_LAT_LONG	0xB030
A661_AC_LONG	0xB020
A661_AC_ORIENTATION	0xB040
A661_ACTIVE_TABBED_PANEL	0xB050
A661_ALPHA_MASK	0xB060

ARINC 661 Parameter Types (16 bits)	
A661_ALTERN_PICTURE_REFERENCE	0xB080
A661_ANIMATION_FLAG	0xB090
A661_ANIMATION_TYPE	**0xB098**
A661_BLINKING_TYPE	0xB0A8
A661_BOUND_X	0xB0B0
A661_BOUND_Y	0xB0C0
A661_BOUND_XY	0xB0D0
A661_BOUND_SIZE_X	0xB0E0
A661_BOUND_SIZE_Y	0xB0F0
A661_BOUND_SIZE_XY	0xB100
A661_BUFFER_OF_PARAM	0xB110
A661_BUFFER_OF_MAPITEM	0xB120
A661_BUFFER_OF_MAPVERT_ITEMS	0xB125
A661_CENTER_X	0xB130
A661_CENTER_XY	0xB150
A661_CENTER_Y	0xB140
A661_COLOR_INDEX	0xB160
A661_CURSOR_POS	0xB170
A661_CURSOR_POS_BYTE	0xB172
A661_ENABLE	0xB180
A661_ENABLE_ARRAY	0xB1A0
A661_ENTRY_ARRAY	0xB190
A661_END_ANGLE	0xB1B0
A661_ENTRY_POP_UP_ARRAY	0xB1C0
A661_ENTRY_VALID	0xB570
A661_EVENT_FLAG	0xB1D0
A661_FILL_INDEX	0xB1E0
A661_FIRST_ACCESS_ENTRY	0xB1F0
A661_FIRST_ACCESS_UA_ENTRY	**0xB1F8**
A661_FIRST_VISIBLE_ENTRY	0xB200
A661_FONT	**0xB590**
A661_FORMAT_STRING	0xB550
A661_FRAME_X	0xB210
A661_FRAME_Y	0xB220
A661_FRAME_XY	0xB230
A661_INNER_RADIUS	0xB240

ARINC 661 Parameter Types (16 bits)	
A661_INNER_STATE_CHECK	0xB244
A661_INNER_STATE_EDIT	0xB248
A661_INNER_STATE_TOGGLE	0xB258
A661_LEGEND_POSITION	0xB260
A661_LINE_LENGTH	0xB270
A661_LOOP_FLAG	**0xB272**
A661_LOOP_TYPE	**0xB273**
A661_MAP_SYNCHRONIZATION_NUMBER	**0xB277**
A661_MASK_REFERENCE	0xB280
A661_MASK_ENABLED	0xB290
A661_NEXT_FOCUSED_WIDGET	**0xB2B8**
A661_NEXT_WIDGET_IDENT	**0xB5B0**
A661_NUMBER_OF_ENTRIES	0xB2A0
A661_NUMBER_OF_VISIBLE_CHILDREN	**0xB5D0**

ARINC 661 Parameter Types (16 bits)	
A661_NUMBER_OF_UA_ENTRIES	**0xB2A8**
A661_NUMERIC_MASK	0xB2B0
A661_ORIENTATION	0xB2C0
A661_OUTER_RADIUS	0xB2D0
A661_PICTURE_ARRAY	0xB2E0
A661_PICTURE_REFERENCE	0xB2F0
A661_POS_X	0xB300
A661_POS_X2	0xB330
A661_POS_X3	0xB360
A661_POS_XY	0xB320
A661_POS_XY2	0xB350
A661_POS_XY3	0xB380
A661_POS_Y	0xB310
A661_POS_Y2	0xB340
A661_POS_Y3	0xB370
A661_PREV_WIDGET_IDENT	**0xB5A0**
A661_PRP_LAT	0xB390
A661_PRP_LAT_LONG	0xB3B0
A661_PRP_LONG	0xB3A0
A661_PRP_SCREEN_X	0xB3C0
A661_PRP_SCREEN_XY	0xB3E0
A661_PRP_SCREEN_Y	0xB3D0
A661_RADIUS	0xB3F0
A661_RANGE	0xB400
A661_ROTATION_ANGLE	0xB410
A661_SCREEN_RANGE	0xB420
A661_SELECTED_ENTRY	0xB430
A661_SHIFT_FIRST_VISIBLE_ENTRY	0xB440
A661_SHUFFLE_TO_FIT_MODE	**0xB5E0**
A661_SIZE_TO_FIT_MODE	**0xB5C0**
A661_SIZE_X	0xB450
A661_SIZE_Y	0xB460
A661_SIZE_XY	**0xB470**
A661_START_ANGLE	0xB480
A661_STRING	0xB490
A661_STRING_ALTERNATE	0xB498

ARINC 661 Parameter Types (16 bits)	
A661_STRING_ARRAY	0xB4A0
A661_STYLE_SET	0xB4B0
A661_SYMBOL_REFERENCE	0xB4C0
A661_TICS_COARSE	0xB4D0
A661_TICS_FINE	0xB4E0
A661_TRANSLATION_X	0xB4F0
A661_TRANSLATION_XY	0xB510
A661_TRANSLATION_Y	0xB500
A661_VALUE	0xB520
A661_VISIBLE	0xB530
A661_VISIBLE_CHILD	0xB540
A661_OPENING_ENTRY	0xB560
A661_BUFFER_OF_FILL_STYLES	0xB0F8
A661_MAJOR_TICK	0xB600
A661_MAPGRID_CELLSIZE	0xB274

ARINC 661 Parameter Types (16 bits)	
A661_MAPGRID_OFFSET	0xB278
A661_MAX_VALUE	0xB610
A661_MIN_VALUE	0xB620
A661_MINMAX_VALUES	0xB580
A661_MINOR_TICK	0xB630
A661_RANGE_X	0xB402
A661_RANGE_Y	0xB403
A661_RANGE_XY	0xB404
A661_PRP_X	0xB3B2
A661_PRP_Y	0xB3B3
A661_PRP_XY	0xB3B4
A661_SOURCE_X	0xB660
A661_TARGET_WIDGET_ID	0xB650
A661_SOURCE_Y	0xB661
A661_SOURCE_DX	0xB662
A661_SOURCE_DY	0xB663
A661_SOURCE_XY	0xB664
A661_SOURCE_DXDY	0xB665
A661_REFRESH	0xB670
A661_SHOW_FAIL	0xB671
Reserved for OEM extensions	0xB800 to 0xBFFF

Table 4.6-9 – Event Types

ARINC 661 Event Types (16 bits)	
A661_EVT_INCREMENT	0xE006
A661_EVT_CURSOR_POS_CHANGE	0xE010
A661_EVT_FIRST_VIS_ENTRY_CHANGE	0xE020
A661_EVT_FRAME_POS_CHANGE	0xE030
A661_EVT_POPUP_CLOSED	0xE040
A661_EVT_SEL_ENTRY_CHANGE	0xE050
A661_EVT_SELECTION	0xE060
A661_EVT_SELECTION_MAP	0xE068
A661_EVT_STATE_CHANGE	0xE070
A661_EVT_STRING_CHANGE	0xE080
A661_EVT_STRING_CHANGE_ABORTED	0xE090
A661_EVT_STRING_CONFIRMED	0xE0A0
A661_EVT_TABBED_PANEL_CHANGE	0xE0B0
A661_EVT_VALUE_CHANGE	0xE0C0
A661_EVT_EDITBOX_OPENED	0xE110
A661_EVT_POPUP_PANEL_CLOSED	0xE120
A661_EVT_ITEM_SYNCHRONIZATION	0xE150
A661_EVT_WATCHDOG_EXPIRED	0xE200
A661_EVT_WATCHDOG_NORMAL	0xE210
A661_EVT_CURSOR_ENTER	**0xE300**
A661_EVT_CURSOR_INSIDE	**0xE310**
A661_EVT_CURSOR_EXIT	**0xE320**

Table 4.6-10 – Boolean Constant Values

ARINC 661 Boolean Constant Values	
A661_FALSE	0x00
A661_TRUE	0x01
A661_TRUE_WITH_VALIDATION	**0x02**

Table 4.6-11 – Integer Constant Values

ARINC 661 Integer Constant Values	
On 8 bits	
A661_UNSELECTED	0x00
A661_EVENT_NONE	0x00
A661_SELECTED	0x01
A661_VALIDATION	0x01
A661_VALIDATION_AND_WHEEL	0x02
A661_WHEEL	0x03
A661_ABSENT	0x10
A661_TOP	0x11
A661_BOTTOM	0x12
A661_LEFT	0x13
A661_RIGHT	0x14
A661_CENTER	0x15
A661_OPEN_UP	0x16
A661_OPEN_CENTERED	0x17
A661_OPEN_DOWN	0x18
A661_UP	0x19
A661_DOWN	0x1A
A661_EDITING	0x1B
A661_ERROR	0x1C
A661_NORMAL	0x1D
A661_NOT_USED	0x00
A661_ITEM_STYLE	0x20
A661_LEGEND	0x21
A661_LEGEND_ANCHOR	0x22
A661_LEGEND_POP_UP	0x23
A661_LINE_ARC	0x24
A661_LINE_SEGMENT	0x25
A661_LINE_START	0x26
A661_SYMBOL_CIRCLE	0x27
A661_SYMBOL_GENERIC	0x28
A661_SYMBOL_ROTATED	0x29
A661_SYMBOL_RUNWAY	0x2A
A661_FILLED_POLY_START	0x2B
A661_SYMBOL_OVAL	0x2C
A661_SYMBOL_TARGET	**0x2D**
A661_TRIANGLE_STRIP_START	**0x2E**
A661_TRIANGLE_SEGMENT	**0x2F**
A661_TRIANGLE_SEGMENT_DOUBLE	**0x31**
A661_TRIANGLE_END	**0x32**
A661_TRIANGLE_END_DOUBLE	**0x33**
A661_TRIANGLE_FAN_START	**0x34**
A661_BOTTOM_CENTER	0x30

ARINC 661 Integer Constant Values	
A661_BOTTOM_LEFT	0x31
A661_BOTTOM_RIGHT	0x32
A661_TOP_CENTER	0x33
A661_TOP_LEFT	0x34
A661_TOP_RIGHT	0x35
A661_BOTTOM_TO_TOP	0x40
A661_LEFT_TO_RIGHT	0x41
A661_RIGHT_TO_LEFT	0x42
A661_TOP_TO_BOTTOM	0x43
A661_ITEM_SYNCHRONIZATION	0x51

ARINC 661 Integer Constant Values	
A661_MDF_BRG_DIST_ACHDG	0x60
A661_MDF_LAT_LONG	0x61
A661_MDF_LEGACY	0x62
A661_MDF_ABSOLUTE	0x65
A661_MDF_RELATIVE	0x66
A661_MDF_DIST_DIST	0x67
A661_MDF_X_DIST	0x68
A661_MDF_Y_ALT	0x69
A661_REPORT_ON_TRANSITION	**0x60**
A661_REPORT_ALL	**0x61**
A661_SIZE_TOP_DOWN	**0x70**
A661_SIZE_BOTTOM_UP	**0x71**
A661_SIZE_LEFT_TO_RIGHT	**0x72**
A661_SIZE_RIGHT_TO_LEFT	**0x73**
A661_NO_SIZING	**0x74**
A661_SHUFFLE_UP	**0x80**
A661_SHUFFLE_DOWN	**0x81**
A661_SHUFFLE_LEFT	**0x82**
A661_SHUFFLE_RIGHT	**0x83**
A661_NO_SHUFFLE	**0x84**
A661_LINE_ARC_INTERACTIVE	0xA4
A661_LINE_SEGMENT_INTERACTIVE	0xA5
A661_LINE_START_INTERACTIVE	0xA6
A661_SYMBOL_CIRCLE_INTERACTIVE	0xA7
A661_SYMBOL_GENERIC_INTERACTIVE	0xA8
A661_SYMBOL_ROTATED_INTERACTIVE	0xA9
A661_SYMBOL_RUNWAY_INTERACTIVE	0xAA
A661_FILLED_POLY_START_INTERACTIVE	0xAB
A661_SYMBOL_OVAL_INTERACTIVE	0xAC
A661_SYMBOL_TARGET_INTERACTIVE	**0xAD**
A661_PIX_FMT_RGBA_8	**0xC0**
A661_PIX_FMT_LUMINANCE_ALPHA_8	**0xC1**

ARINC 661 Integer Constant Values	
A661_PIX_FMT_COLOR_INDEXED_8	**0xC2**
(0xC3 through 0xCF are reserved for future use)	
A661_DONT_RUN	**0x00**
A661_RUN	**0x01**
A661_RUN_ONCE	**0x02**
A661_LOOP_FORWARD	**0x00**
A661_LOOP_FORWARD_AND_RESET	**0x01**
A661_LOOP_FORWARD AND_BLANK	**0x02**
A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK	**0x03**
A661_EDB_CHANGE_CONFIRMED	0x00
A661_EDB_ALL_CHANGE	0x01
A661_EDB_OPEN_CLOSE	0x02
On 16 bits	
A661_STYLE_SET_DEFAULT	0x0000

5.1 Overview

Symbol widget and the SYMBOL_GENERIC and SYMBOL_ROTATED map items refer to symbols by a “SymbolReference” ID number. A CDS supports a set of predefined symbols. The existing predefined CDS symbology can also be supplemented by symbols defined within DFs. The DF Symbol definitions are found near the start of the DF, before any of the UA Layer definitions. Any UA Layers of a DF can reference the symbols defined in that DF.

The following scheme should be used to find the appropriate symbol for a given symbol ID number:

- If a Symbol with matching ID is found in the DF, use it.
- Otherwise, use the predefined symbol with that ID.

In the case of the SYMBOL_GENERIC and SYMBOL_ROTATED map items, the legend anchor command of a symbol defines the first legend position.

This section describes the Symbol Definition Commands that are used within the Symbol Definition Structure to specify the graphical appearance of the symbol.

- See Section 4.5.3 for the format of the Definition Time structure in general, and the Symbol Definitions Structure in particular.

5.2 Symbol Definition Commands

A single Symbol Definition contains a sequence of Symbol Definition Commands. A Symbol Definition can have up to **four** different representations, each consisting of graphic primitive and/or attribute commands:

- The standard representation begins the symbol definition.
- An optional focus representation begins with a FOCUS command, and follows the standard representation.
- An optional highlight representation is last, beginning with a HIGHLIGHT command.
- **An optional sensitive area representation to define the interactive zone of the symbol.**

Commands given in one representation (standard, focus, highlight **or sensitive area**) of a symbol do not affect commands given in another representation of that symbol (or any other symbol).

symbol_definition ::=
symbol_representation
[FOCUS_Command *symbol_representation*]
[HIGHLIGHT_Command *symbol_representation*]

[RECT_AREA_Command | CIRC_AREA_Command]

symbol_representation ::=
{ symbol_attribute_command **|** symbol_graphic_primitive_command } +

The symbol attribute commands and symbol graphic primitive commands are described in sections 5.2.2 and 5.2.3 respectively.

5.2.1 Top Level Commands

5.2.1.1 Focus

The FOCUS command identifies the beginning of the definition of the focus representation of the symbol.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_FOCUS
UnusedPad	N/A	16	

5.2.1.2 Highlight

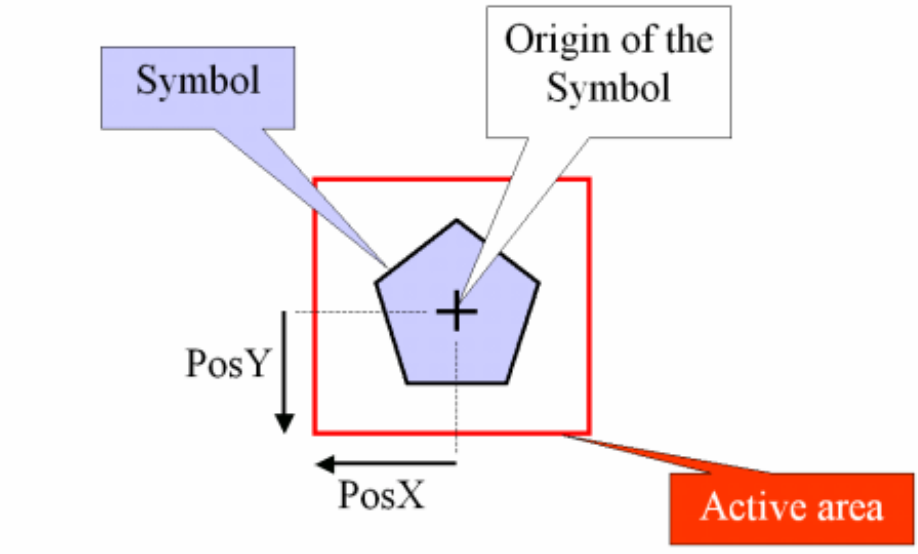
The HIGHLIGHT command identifies the beginning of the definition of the highlight representation of the symbol.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_HIGHLIGHT
UnusedPad	N/A	16	

5.2.1.3 Sensitive Area

The Sensitive Area command identifies the definition of the interactive zone representation of the symbol. This zone is a rectangular one or a circular one. If this zone is not defined, the behavior is OEM dependent (the symbol may be declared as not interactive, or a default interactive zone may be defined by the CDS).

For a rectangular zone, the parameter "RotationAllowed" is set to A661_TRUE if the Sensitive Area is rotated with the symbol in case of A661_SYMBOL_ROTATED. If this parameter is set to A661_FALSE, the Sensitive area rectangle is always parallel to X and Y axis.



Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_RECTANGULAR
RotationAllowed	N/A	8	A661_TRUE, A661_FALSE
UnusedPad	N/A	8	
PosX	long	32	The X position : it is the bottom left corner of the rectangle, relative to the symbol origin
PosY	long	32	The Y position : it is the bottom left corner of the rectangle, relative to the symbol origin
SizeX	ulong	32	The width of the Sensitive area.
SizeY	ulong	32	The height of the Sensitive area.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_CIRCULAR
UnusedPad	N/A	16	
PosX	long	32	The X position of the center of the circle
PosY	long	32	The Y position of the center of the circle
Radius	ulong	32	The radius of the circle.

5.2.2 Symbol Attribute Setting Commands

Symbol Attribute Setting Commands set graphical attributes (Color, Linestyle, Font, Halo) that affect Graphic Primitive Commands.

The StyleSet and/or Color and/or Halo properties of the widget or Map Item that references the symbol determine the initial attribute settings of each of the representations of the symbol. Attribute setting commands (if any) made within a representation of the symbol definition override those initial defaults. The attribute setting commands made within a symbol definition only apply within the context of that symbol: these settings cannot affect subsequent widgets (including other symbols) or map items.

5.2.2.1 Set Color

The SET_COLOR command sets the current color index, affecting the color of lines, boundary lines and fills. It is an index into the same color table referenced by color index properties of Gp widgets.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _SET_COLOR
ColorIndex	uchar	8	Index into a CDS-defined color table.
UnusedPad	N/A	8	

5.2.2.2 Set Line Style

The SET_LINE_STYLE command sets (non-color) attributes that effect line (and line segment) drawing: for example pattern and width. The style is an index into a line style table stored on the CDS. Linestyle modulation is reset with every SET_LINE_STYLE command.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _SET_LINE_STYLE
Style	ushort	16	Index into a CDS-defined line style table.

5.2.2.3 Set Font

The SET_FONT command sets the current font to the given font index.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _SET_FONT
Font	uchar	8	Index into a CDS-defined font table.
UnusedPad	N/A	8	

5.2.2.4 Set Halo

The SET_HALO command sets the current halo setting.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _SET_HALO
Halo	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	

5.2.3 Graphic Primitive Commands

This section describes symbol commands to define the graphic primitives that make up a symbol.

All coordinate, height, width and radius arguments are in units of hundredths of mm. All angles are in degrees and follow the rules specified in section 2.3.4.2, Angles. The (0,0) position of the symbol represents the part of the symbol that will be placed at the widget or map item's position. Symbols are sensitive to rotation and/or translations that are applied by ancestor widgets.

5.2.3.1 Legend Anchor

The LEGEND_ANCHOR command explicitly sets the anchor position for the legend attached to the symbol representation. If there is more than one Legend Anchor command, the behavior is OEM-dependent. If there is no LEGEND_ANCHOR command within a symbol representation, the legend anchor position is (0,0). The LEGEND_ANCHOR command does not correspond to any Gp widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_LEGEND_ANCHOR
UnusedPad	N/A	16	
PosX	long	32	The X position.
PosY	long	32	The Y position.

5.2.3.2 Arc Ellipse

The ARC_ELLIPSE command defines an arc shape that may be a portion of an ellipse or an arc. It is defined by a bounding box where a rectangle is specified and the ellipse is drawn touching the rectangle. When the bounding box is a square, the arc will be a circle. The major and minor axes of the ellipse are implicitly along the cardinal directions of the bounding box.

The ARC_ELLIPSE command draws the same shape as the GpArcEllipse widget. As in the GpArcEllipse widget, the ARC_ ELLIPSE becomes a complete ellipse or circle when the StartAngle and EndAngle represent the minimum and maximum

possible values of a 32-bit fr(180), corresponding to –180 degrees and slightly less than +180 degrees respectively.

The ARC_ ELLIPSE is drawn using the current color (in both the filled and unfilled cases). If it is unfilled, the outline is drawn using the current linestyle.

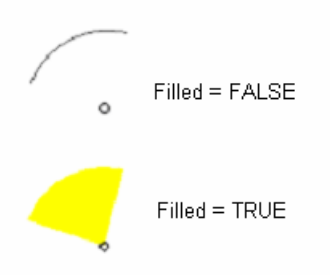
Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_ARC_ELLIPSE
Filled	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	
PosX	long	32	The X start position of the bounding box (lower left corner).
PosY	long	32	The Y start position of the bounding box (lower left corner).
SizeX	ulong	32	The maximum width the ARC_ELLIPSE can have (if it's StartAngle and EndAngle defines a full ellipse).
SizeY	ulong	32	The maximum height the ARC_ELLIPSE can have (if it's StartAngle and EndAngle defines a full ellipse).
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the ARC_ELLIPSE.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the ARC_ELLIPSE.

5.2.3.3 Arc Circle

The ARC_CIRCLE command can define either an arc or a circle. It is defined by a center position, a start angle and an end angle. The arc sweeps counterclockwise from the start angle to the end angle.

The ARC_CIRCLE command draws the same shape as the GpArcCircle widget.

The following figure illustrates the two ARC_CIRCLE command cases, depending on the Filled setting. The small circles are for reference purposes and indicate the circle centers (they don't actually get drawn). The ARC_CIRCLE is drawn using the current color (in both the filled and unfilled cases). If it is unfilled, the outline is drawn using the current linestyle.



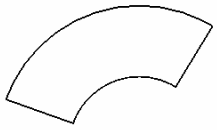
Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _ARC_CIRCLE
Filled	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	
PosX	long	32	The center X position of the arc or circle.
PosY	long	32	The center Y position of the arc or circle.
Radius	long	32	The radius of the arc or circle.
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the arc or circle.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the arc or circle.

5.2.3.4 Crown

The CROWN command defines a 2D doughnut-shaped region, defined by a center, two radii, a StartAngle and an EndAngle. The crown sweeps the region going counterclockwise from the start angle to the end angle. The crown becomes closed to form a complete “doughnut” shape when the StartAngle and EndAngle represent the minimum and maximum possible values of a 32-bit fr(180), corresponding to –180 degrees and slightly less than +180 degrees respectively.

A crown can be filled or unfilled (outlined). The following diagram shows an outlined crown. The crown is drawn using the current color (in both the filled and unfilled cases). If it is outlined, the outline is drawn using the current linestyle.

The CROWN command draws the same shape as the GpCrown widget.



Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _CROWN
Filled	uchar	8	If set to True, the Crown will be filled. If False, an outline of the Crown will be drawn.
UnusedPad	N/A	8	
PosX	long	32	The center X position of the crown.
PosY	long	32	The center Y position of the crown.
InnerRadius	ulong	32	The inner radius.
OuterRadius	ulong	32	The outer radius.
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the crown
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the crown.

5.2.3.5 Line

The LINE command draws a line between the two positions specified. The LINE is drawn using the current color and linestyle. The LINE command draws the same shape as the GpLine widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _LINE
UnusedPad	N/A	16	
PosXStart	long	32	The start X position of the line.
PosYStart	long	32	The start Y position of the line.
PosXEnd	long	32	The end X position of the line.
PosYEnd	long	32	The end Y position of the line.

5.2.3.6 Line Polar

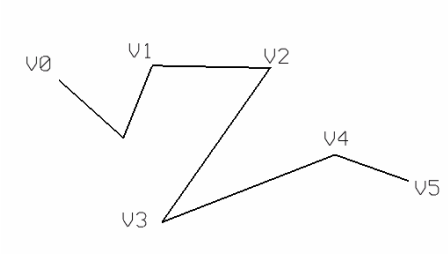
The LINE_POLAR command defines a line in polar coordinates with an X,Y coordinate start position, a line length, and a draw angle. The LINE_POLAR is drawn using the current color and linestyle. The LINE command draws the same shape as the GpLinePolar widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_LINE_POLAR
UnusedPad	N/A	16	
PosXStart	long	32	The start X position of the line.
PosYStart	long	32	The start Y position of the line.
RotationAngle	fr(180)	32	The angle at which the line is drawn.
LineLength	long	32	The length of the line.

5.2.3.7 Polyline

The POLYLINE command defines a sequence of connected lines. It is drawn using the current color and linestyle. The POLYLINE command does not correspond to any Gp widget.

The following figure shows an example of how an unclosed polyline defined using vertices (V0 .. V5) could be defined. (When drawn, the polyline will not show the (V0..V5) labels).



Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _POLYLINE
NumVertices	ushort	16	3 <= number of vertices
Closed	uchar	8	A661_TRUE: an extra line is drawn from the last vertex to the first vertex. A661_FALSE: no extra line is drawn from the last vertex to the first vertex.
UnusedPad	N/A	24	
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.3.8 Rectangle

The RECTANGLE command defines a rectangle, according to a lower left corner position, a height and a width. The RECTANGLE is drawn using the current color (in both the filled and unfilled cases). If unfilled, it uses the current linestyle. The RECTANGLE command draws the same shape as the GpRectangle widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _RECTANGLE
Filled	uchar	8	If set to True, the Rectangle will be filled. If set to False, the Rectangle will be outlined.
UnusedPad	N/A	8	
PosX	long	32	The X start position of the rectangle (lower left corner).
PosY	long	32	The Y start position of the rectangle (lower left corner).
SizeX	ulong	32	The width of the rectangle.
SizeY	ulong	32	The height of the rectangle.

5.2.3.9 Triangle

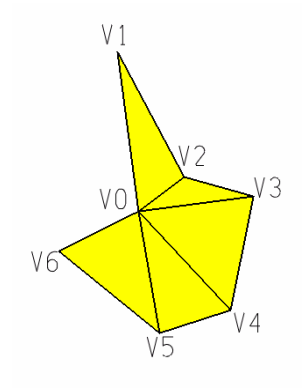
The TRIANGLE command defines a triangle, according to three defined vertices. The TRIANGLE is drawn using the current color (in both the filled and unfilled cases). If unfilled, it uses the current linestyle. The TRIANGLE command draws the same shape as the GpTriangle widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _TRIANGLE
Filled	uchar	8	A661_TRUE: the Triangle will be filled. A661_FALSE: the Triangle will be outlined.
UnusedPad	N/A	8	
PosX	long	32	The X start position of the triangle.
PosY	long	32	The Y start position of the triangle.
PosX2	long	32	The X position of the second vertex of the triangle
PosY3	long	32	The Y position of the second vertex of the triangle
PosX3	long	32	The X position of the third vertex of the triangle
PosY3	long	32	The Y position of the third vertex of the triangle

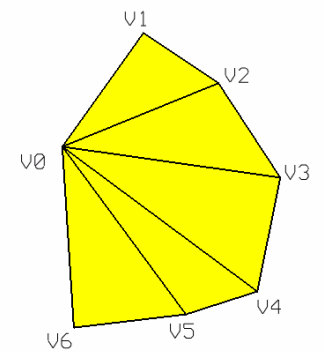
5.2.3.10 Triangle Fan

The TRIANGLE_FAN command defines a filled shape composed out of a fan of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the first and last vertices of the previous triangular section. At least three vertices must be specified. Any convex polygon can be represented as a triangle fan, by just specifying its vertices in the natural order. A triangle fan is not necessarily a convex polygon, though. It is drawn using the current color. The TRIANGLE_FAN command does not correspond to any Gp widget.

The following figure shows how the Vertices (V0 ..V6) of a triangle fan define a filled shape formed out of triangles. When drawn, a triangle fan does not draw the lines or vertex labels shown in this figure. In this case it defines a concave polygon:



The following figure illustrates a convex polygon case:

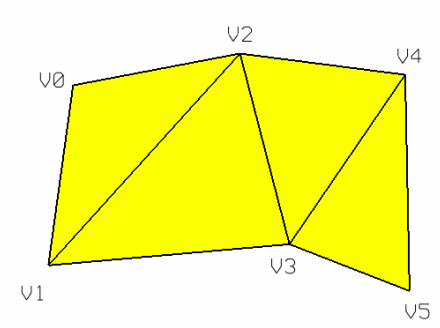


Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _TRIANGLE_FAN
NumVertices	ushort	16	Number of vertices, must be >= 3.
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.3.11 Triangle Strip

The TRIANGLE_STRIP command defines a filled shape composed out of a linked strip of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the last two vertices of the previous triangular section. It is drawn using the current color. The TRIANGLE_STRIP command does not correspond to any Gp widget.

The following figure shows how the Vertices (V0 ..V5) of a triangle strip define a filled shape formed out of triangles. When drawn, a triangle fan does not draw the lines or vertex labels shown in this figure.



Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _TRIANGLE_STRIP
NumVertices	ushort	16	Number of vertices, must be >= 3.
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.3.12 Text

The TEXT command defines a fixed text string that is part of symbol and is drawn using the current font. The first character of the text string is placed at the specified position and subsequent characters are drawn to the right. It is drawn using the current color. The effect of current linestyle on Text for a particular font is OEM-dependent. The TEXT command does not correspond to any Gp widget.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _TEXT
Alignment	uchar	8	specifies the alignment of the text with respect to PosX and PosY. TOP_LEFT TOP_CENTER TOP_RIGHT LEFT CENTER RIGHT BOTTOM_LEFT BOTTOM_CENTER BOTTOM_RIGHT
UnusedPad	N/A	8	
PosX	long	32	The X position.
PosY	long	32	The Y position.

Field Name	Type	Size (bits)	Description
RotationAngle	fr(180)	32	Specifies rotation of text with respect to PosX and PosY.
StringText	string	8 * string length + pad	Null-terminated string, followed by zero, one, two or three extra NULL for 32 bit alignment.

5.2.3.13 Size

The SIZE command defines the size of a symbol representation. If multiple SIZE commands are found in a representation, all but the last are ignored. If there is no SIZE command in a symbol representation then a default size of zero is used for that representation.

Size defines a bounding box which does not change if the symbol is rotated. Unlike other objects having SizeX/SizeY parameters, symbols are not clipped to the SizeX/SizeY region defined by this command.

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN _SIZE
UnusedPad	N/A	16	
SizeX	ulong	32	The width of the symbol representation.
SizeY	ulong	32	The height of the symbol representation.

6.1 Introduction

This section describes how an ARINC 661 definition file is defined in XML format. It is assumed that the reader is familiar with XML [1] and ARINC 661 widgets.

The XML DF format is intended to provide a convenient interchange format between ARINC 661 tools. It also has the advantage that it is human-readable. The XML DF format includes all the information found in the binary DF, but may also contain additional information. For example, the XML DF optionally allows names to be given to widgets (in addition to the widget id). This is convenient for various ARINC 661 tools (e.g. UA Layer editors). There is also provision for additional OEM-specific information to be added (which also might not correspond to information found in the binary DF).

Since the XML DF can contain more information than a binary DF, a round-trip conversion from an XML DF to a binary DF and back again could lose information.

The XML DF grammar description provided defines the *core grammar subset*. It does not describe the full grammar because it is expected that OEM-specific XML elements will also be added.

The XML DF grammar is not widget-type-specific, so does not need to be changed to support new widgets.

Refer to Appendix C for XML DF examples.

6.2 Description

ARINC 661 Definition XML data consists of the following declarations, in the order given below. XML comments (starting with "<!--" and ending in "-->") may also be given, but not before the "<?xml" element).

1. **An XML version processing instruction. The "encoding" attribute should specify UTF-8 encoding as defined in ISO 10646-1:2000. This is the XML default value for the encoding attribute, so this specification is optional. There are some additional character restrictions in the ARINC 661 XML file, since the ARINC 661 character set does not correspond to UNICODE. See Section 6.2.5 for more details. For example:**

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   ```

2. **A Document Type Declaration (DTD) specification, as found in a "DOCTYPE" element. There are different formats of this element, some which declare the grammar locally within the same file, and others which refer to an external file. The example below refers to an external DTD file. The root type (always the first argument of the DOCTYPE) must be a661_df. The DTD does not have to be exactly as defined in Section 6.3 of this specification (for example, it could introduce new XML element tags), but needs to follow the Rules given below in this section.**

   ```
   <!DOCTYPE a661_df SYSTEM "a661.dtd">
   ```

3. **A top-level a661_df XML element, which defines the hierarchy of the Definition File via its nested children.**

Rules for an ARINC 661 XML DF to be compliant with a particular version of ARINC 661:

1. It should have a DTD (as specified by the DOCTYPE element), and be both well-formed and valid (i.e. it has to conform to that DTD).
2. Its DTD can only differ from the DTD specified in section 6.3 in the following ways :
   - **It can introduce new "<!ELEMENT" and/or "<!ATTLIST" and/or "<!ENTITY>" commands to define new XML elements.**
   - **It can modify existing <!ELEMENT" commands to introduce new XML elements as possible children of existing XML elements.**
   - **It can modify existing "<!ATTLIST" commands to add new attributes to existing XML elements.**
   - **It can have other syntactic differences that don't affect content. This includes (but is not limited to) differences in comments or whitespace.**

A tool loading the ARINC 661 XML DF:

- **should reject any file that is not both "valid" and "well-formed" in the standard XML sense. In other words, the file has to have good XML syntax and must conform to its DTD to be accepted.**
- **has undefined behavior if the ARINC 661 XML DF is not ARINC 661 compliant, as defined in the rules above.**
- **should skip any of the following, possibly giving warnings :**

1. any unrecognized XML element (including children).
2. any unrecognized XML attribute of a standard ARINC 661 XML element.
3. any standard A661 XML element that it does not support (including children). Here are examples:
   - **any property XML element (i.e. prop, structprop, or arrayprop) whose (property) name attribute value isn't supported, as well as any children it might have.**
   - **any symboldefncmd XML element whose type attribute value is not supported, as well as any children it might have. (see section 6.2.1).**
   - **any a661_widget XML element whose type attribute value is not supported, as well as any children it might have. (see section 6.2.2).**

The following subsections specify the **standard** content of the top-level XML element.

The following table describes a table format that is used in the remainder of this section to describe each XML element:

XML Element	***name of the XML Element***
Description	*description of the XML Element*
XML Attributes	*List of the XML attributes of the XML Element.* • *i.e. the names of the attributes that are specified as "attrib=value" pairs in the XML element.*
Contains (in order)	*A description of the legal XML element children of this XML element, if any.*

XML Element	*name of the XML Element*
Properties	*If the XML element allows a **model** child XML element, that **model** element can specify properties. For example, properties are used to specify parameters of an **a661_widget** XML element. See section 6.2.3 for details regarding the **model** XML element.*

The following XML fragment gives an example of how **model** XML elements are used to define properties. In it a **model** XML element specifies the parameter values of a CheckButton widget:

```
<a661_widget name="CKB_CHOICE" type="A661_CHECK_BUTTON">
     <model>
       <prop name="WidgetId" value="2"/>
       <prop name="Visible" value="A661_TRUE"/>
       <prop name="Enable" value="A661_TRUE"/>
       <prop name="CheckButtonState" value="A661_UNSELECTED"/>
       <prop name="StyleSet" value="1"/>
       <prop name="PosX" value="100"/>
       <prop name="PosY" value="100"/>
       <prop name="SizeX" value="3000"/>
       <prop name="SizeY" value="900"/>
       <prop name="NextFocusedWidget" value="0"/>
       <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
       <prop name="LabelSring" value="SELECT"/>
       <prop name="MaxStringLength" value="7"/>
       <prop name="Alignment" value="A661_LEFT"/>
       <prop name="PicturePosition" value="A661_LEFT"/>
    </model>
 </a661_widget>
```

The following table describes the **a661_df** element, which is the root element of the XML data. The following parts of this element provide information that is not found in the binary DF:

- **name** XML attribute.

XML Element	a661_df
Description	This is the root object of the XML data. It defines the entire DF.
XML Attributes	• **name** (optional, string): this is the name of the DF. It corresponds to the name of the file (not including file extension). • **library_version** (required, unsigned char): this gives the implementation dependent library version of the DF as found in the binary DF header. Refer to section 4.5.3.2. • **supp_version** (required, unsigned char): this gives the supplement version of the DF as found in the binary DF header. Refer to section 4.5.3.2.
Contains (in order)	• A **model** element. • An optional **symboltable** element • Zero or more **picturetable** elements. • One or more **a661_layer** elements
Properties	• **ApplicationId** (unsigned short) : this is the identification number of the user application associated with this DF. This information is found in the binary DF header. Refer to section 4.5.3.2.

6.2.1 Picture and Symbol Graphical Definitions

This section describes XML elements related to Symbol Graphical Definitions **and Picture Definitions** that are defined locally within the DF**. S**ee section 5 for more information on **Symbol Definitions**. **See** S**ection 7 for more information on Picture Definitions.**

The following table describes the symboltable XML element.

XML Element	symboltable
Description	Defines a table of Symbol Graphical Definitions that are stored within the DF.
XML Attributes	None
Contains	**Zero** or more **symboldefn** elements.
Properties	None.

The following table describes the **symboldefn** XML element. The following parts of this element provide information that is not found in the binary DF:

- **name** XML attribute.

XML Element	symboldefn
Description	Defines the graphical representation of a symbol.
XML Attributes	name (optional) : a name for the symbol definition. This name might be referenced elsewhere in the XML file instead of a numeric symbol id value.
Contains (in order)	• a required **model** element • required **stdrepr** element. • an optional **focusrepr** element • an optional **highlightrepr** element • an optional **rectangular_sensitive_area or circular_sensitive_area** element
Properties	Id : an unsigned short value giving the id number of the symbol.

The following table describes the **stdrepr** XML element.

XML Element	stdrepr
Description	Defines the standard representation of a symbol defined within the DF.
XML Attributes	None
Contains	**Zero** or more **symboldefncmd** elements.
Properties	None.

The following table describes the **focusrepr** XML element.

XML Element	focusrepr
Description	Defines the focus representation of a symbol defined within the DF.
XML Attributes	None
Contains	**Zero** or more **symboldefncmd** elements.
Properties	None.

The following table describes the **highlightrepr** XML element.

XML Element	highlightrepr
Description	Defines the highlight representation of a symbol defined within the DF.
XML Attributes	None
Contains	**Zero** or more **symboldefncmd** elements.
Properties	None.

The following table describes the **rectangular_sensitive_area** XML element.

XML Element	rectangular_sensitive_area
Description	Defines a **rectangular sensitive** area definition command.
XML Attributes	**None**
Contains	One model element.

Properties	The properties available correspond directly to the parameters defined ~~per~~ **in the rectangular sensitive area** definition command in section 5.

The following table describes the circular_sensitive_area XML element.

XML Element	**circular_sensitive_area**
Description	**Defines a circular sensitive area definition command.**
XML Attributes	**None**
Contains	**One model element.**
Properties	**The properties available correspond directly to the parameters defined in the circular sensitive area definition command in section 5.**

The following table describes the **symboldefncmd** XML element. The following parts of this element provide information that is not found in the binary DF:

- **name** XML attribute.

XML Element	**symboldefncmd**
Description	Defines a symbol graphical definition command.
XML Attributes	• **name** (optional) : this gives the symbol definition command object a name. This name would be purely for descriptive purposes, and would not be referenced elsewhere in the XML file. • **type** (required) : this gives the type of symbol graphical definition command (e.g. A661_SYMBOL_DEFN_ARC_ELLIPSE). Refer to section 5 for the legal symbol graphical definition command types. Note: the following symbol definition types cannot be used here, since they are handled via the **rectangular_sensitive_area, circular_sensitive_area, focusrepr** and **highlightrepr** XML elements: • **A661_SYMBOL_DEFN_RECTANGULAR** • A661_SYMBOL_**DEFN_CIRCULAR** • A661_SYMBOL_DEFN_FOCUS • A661_SYMBOL_DEFN_HIGHIGHT
Contains	One **model** element.
Properties	The properties available correspond directly to the parameters defined per type of symbol definition command in section 5. Unused pad parameters should not be specified as properties.

The XML DF defines pictures a bit differently than the binary DF. Refer to Section 7 for more details on the binary representation. What follows is a quick comparison.

The binary DF has zero or more A661_Picture_Block_Structure_DT structures, each of which specifies:

- **PixelFormat: A661_PIX_FMT_RGBA_8, A661_PIX_FMT_LUMINANCE_ALPHA_8 , A661_PIX_FMT_COLOR_INDEXED_8 or OEM-specific value.**
- **NumberOfPicturesInBlock.**
- **NumberOfColorTableEntries.**
- **ColorTableFormat: A661_PIX_FMT_RGBA_8, A661_PIX_FMT_LUMINANCE_ALPHA_8 , or OEM-specific value. (Only useful if NumberOfColorTableEntries is greater than 0 and the PixelFormat references a color palette, e.g. A661_PIX_FMT_COLOR_INDEXED_8).**

- **The color table, composed of NumberOfColorTableEntries color table entries, whose representation depends on the ColorTableFormat.**
- **The picture definitions, each of which specifies:**
  - **PictureReference.**
  - **NumberOfPixelsWidth.**
  - **NumberOfPixelsHeight.**
  - **The pixel data of the picture according to the PixelFormat (and references the color table if applicable).**

The XML DF has an optional picturetable XML element, which contains:

- **Zero or more picturedefn XML elements, each of which specifies:**
  - **A PictureReference property.**
  - **A PixelFormat property.**
  - **A ColorTableFormat property, only used if PixelFormat is color palette-based (e.g. A661_PIX_FMT_COLOR_INDEXED_8).**
  - **An ImageFile property giving the name of a PNG file.**

When converting from an XML DF to a binary DF:

- **The pixel data is found in the image file. It might need to be transformed before copying to the binary DF, e.g. if it isn't represented in the required PixelFormat.**
- **The NumberOfPixelsWidth and NumberOfPixelsHeight information is found in the image file.**
- **The color tables are computed as necessary based on the required ColorTableFormat and the colors used in the image files.**
- **A661_Picture_Block_Structure_DT structures are created where necessary, from picturedefn XML elements having the same PixelFormat (and ColorTableFormat if applicable). The maximum size of the color tables is limited by the PixelFormat . This may limit which pictures may be grouped together. For example, pictures using PixelFormat of A661_PIX_FMT_COLOR_INDEXED_8 can only be grouped together if the total number of colors they use does not exceed 256. The exact details of this algorithm are OEM-specific. The main goal is to reduce binary DF size by grouping pictures using the same color palette format together, where possible.**

The following table describes the picturetable XML element, defining the set of picture definitions stored in the DF. A picturetable corresponds to the sequence of PictureBlockStructure blocks found near the top of the binary DF.

XML Element	**picturetable**
Description	Defines a block of picture definitions that are stored within the DF.
XML Attributes	None
Contains	Zero or more **picturedefn** elements.
Properties	None

The following table describes the **picturedefn** XML element. The following parts of this element provide information that is not found in the binary DF:

- **ImageFile** property.

The **ImageFile** property refers to an external PNG (Portable Network Graphics) file which contains the picture pixel data. The PNG file format supports a variety of pixel formats, including both (RGBA) palette-based and RGBA. It also has the advantages that it supports lossless compression and is open source. Refer to *www.w3.org/Graphics/PNG/* or *www.libpng.org* for details.

XML Element	picturedefn
Description	Defines a local picture in a DF. A PNG (Portable Network Graphics) file is used to store the pixel data and image size. The PNG file format supports lossless compression and a variety of pixel formats.
XML Attributes	• name (optional): this is purely for descriptive purposes.
Contains	One model element.

Properties	• PictureReference: ushort: a numeric identifier that widgets such as Picture and PicturePushButton use to identify a bitmap they wish to display.
	• ImageFile (string): file pathname specifying a PNG (Portable Network Graphics) file that contains the image contents and specifies the image size in pixels. The path may be relative or absolute. Relative pathnames are relative to the directory which contains the XML DF file containing the a661_picture_defn element. Note: it is recommended to avoid the usage of absolute pathnames to help make it easier to move XML files from one directory (or machine) to another.
	• PictureFormat: (uchar): indicates how the picture pixel information will be stored in the binary DF. It does not necessarily correspond to the internal representation of the PNG file itself. The most common PNG internal representations are RGBA-based. It is possible to use an RGBA PNG file in conjunction with any of the picture representations below. In some cases processing could be required to transform the PNG format into the requested binary DF format.
	  • A661_PIX_FMT_RGBA_8 : It is easy to extract this format of RGBA pixel information from a PNG file, regardless of its internal representation.
	  • A661_PIX_FMT_LUMINANCE_ALPHA_8 : PNG supports a "grayscale with alpha" representation that maps to this directly. If the PNG is in another format that allows non-gray colors its pixel values would be converted to grayscale values.
	  • A661_PIX_FMT_COLOR_INDEXED_8 : This setting requires that the PNG file uses 256 or less different colors.
	• ColorTableFormat (uchar) : specifies the format of the color table in the binary DF (only useful if PictureFormat is a color palette format such as A661_PIX_FMT_COLOR_INDEXED_8). The PNG file could have a different representation (e.g. a non-palette representation).
	  • A661_PIX_FMT_RGBA_8 : There are no warning or error cases with this choice.
	  • A661_PIX_FMT_LUMINANCE_ALPHA_8 : If the PNG is in another format that allows non-gray colors its pixel values would be converted to grayscale values.

6.2.2 Layers and Widgets

The following table describes the **a661_layer** XML element. The following aspects of this element provide information that is not found in the binary DF:

1. **name** XML Attribute
2. **Height** property
3. **Width** property

XML Element	a661_layer
Description	Defines a layer.
XML Attributes	• **name** (optional) : this is purely for descriptive purposes.
Contains	• A **model** element. • Zero or more **a661_widget** elements.
Properties	• **LayerId** (unsigned **char**) : the layer id number. • **ContextNumber** (unsigned short) : the "context number" of the layer. • **Height** (unsigned long) : the height of the layer, in 1/100 mm. • **Width** (unsigned long) : the width of the layer, in 1/100 mm.

The following table describes the a661_widget XML element. The following aspects of this element provide information that is not found in the binary DF:

- name XML Attribute

XML Element	a661_widget
Description	Defines a widget.
XML Attributes	• **name** (optional) : this is purely for descriptive purposes. • **type** (required) : this gives the type of the ARINC 661 widget (e.g. "A661_PUSHBUTTON").
Contains (in order)	• A **model** element. • Zero or more **a661_widget** elements (which represent the child widgets of a container widget).
Properties	There is a property specified for each widget "D" or "DR" property, **with the following exceptions:** • **ParentIdent** parameters are not specified using a property because they can be calculated based on the context: 0 if is a top-level widget within the layer, otherwise it is the ParentIdent of the parent widget. • **WidgetType** parameters are not specified using a property since they are already specified using the **type** XML Attribute described above. • **Unused** pad locations are not specified using properties. Note: all unused pad locations are implicitly zero-filled.

6.2.3 Properties

Property values are specified **using the model XML element**.

It isn't applicable in this section to do either of the following:

- Provide the Properties row in the XML element tables. Property values don't have properties.
- Indicate which parts of these elements are found in the binary DF. Look at the Properties sections of the tables in the sections above for this information.

Here is a summary of the XML elements described in this section:

The **model** XML element holds the property values of a particular object.

- The prop, structprop, and arrayprop XML elements specify the value of a property, having a simple, structure or array type, respectively
- The field, structfield and arrayfield XML elements specify the value of a field of a structure, having a simple, structure or array type, respectively
- The entry, xyentry, structentry, and arrayentry XML elements specify the value of an entry of an array, having a simple, (x,y) coordinate, structure or array type, respectively

In principle these elements can be nested deeply, to allow arrays of structures of arrays, etc. In practice the usage is much more limited, based on the much simpler data types which are used within the ARINC 661 standard.

If an object has a property, but its value is not specified within the model **element**, an OEM-specific default value of that property should be used instead.

Rules for handling string values:

- Properties that contain a variable number of elements should be modeled using an array. The property that specifies the size of the array should be specified before the array property itself. Here are some examples:
  - The EntryList parameter of the ComboBox widget
  - The PopUpIdentArray, EnableArray and StringArray parameters of the PopUpMenu or PopUpMenuButton widgets
- Properties that specify an enumerated type should use the name of the enumerated values in the XML file. For example, Boolean properties should show values “A661_TRUE” and “A661_FALSE” in the XML file.
- Properties that provide an attribute index (color, styleset, symbol or linestyle) can be given either as a symbolic name, or as a numeric index. The symbolic names are OEM-specific.
  - In the case of a symbol index, if it refers to a locally defined symbol, the name of the symbol as given in the symboldefn XML item can be used. The name of a CDS-defined OEM-specific symbol could also be used.
- Properties that specify an “fr” (fixed real) value can be expressed in either of the following ways
  - as a “real” value if the number given contains a “.”
  - as an integral representation of the fixed real value if the number given does not contain a “.”
- For integral numeric values, a C-style hex representation should also be supported. A hex value starts with zero, followed by the letter “x” (lowercase or uppercase), followed by a sequence of hex digits (e.g. “0x6F12”).
- **ARINC 661 characters are restricted to have codes between 0 and 255 inclusive. The UTF-8 XML encoding used in ARINC 661 XML files can in general support all of UNICODE (whose codes can be much larger than 255), via the use of multi-byte characters. The ARINC 661 XML file does therefore not support full UTF-8. Some ARINC 661 characters do not conform to UNICODE. Refer to section 3.2.5.1 for the description of the**

ARINC 661 character set, which introduces ARINC 661 specific characters, and also leaves many characters OEM-dependant.

Note: Characters with codes 0 to 127 are the same in ASCII and UNICODE.

- **Any character - even non printable - can be described in the XML file through its character code. The selected syntax is &#xhh; where hh is the 2 hex digit character code. Note: hex letter digits 'A' .. 'F' must be given in uppercase. Therefore B stands for B and ABC stands for ABC. Non-printable characters, and characters which do not conform to UNICODE should be specified in this way. This avoids problems in XML editors and viewers.**

The following table describes the **model** XML element:

XML Element	**model**
Description	Contains the property settings of an object. The "object" is not necessarily only a widget. Refer to all XML elements that specify "Properties".
XML Attributes	None
Contains (in order)	Zero or more **prop** and/or **structprop** and/or **arrayprop** elements. (Each child specifies the value of a property).

The following table describes the **prop** XML element:

XML Element	**prop**
Description	Sets a simple property value.
XML Attributes	• **name** (required) : the name of the property. • **value** (required) : the simple value of the property.
Contains	Has no children.

The following table describes the **structprop** XML element:

XML Element	**structprop**
Description	Sets a struct-valued property value.
XML Attributes	• **name** (required) : the name of the property.
Contains	One or more **field** and/or **structfield** and/or **arrayfield** elements. (Each child element specifies the value of a field of the structure property)

The following table describes the **arrayprop** XML element:

XML Element	**arrayprop**
Description	Sets an array-valued property value.
XML Attributes	• **name** (required) : the name of the property.
Contains	zero or more **entry** and/or **xyentry** and/or **structentry** and/or **arrayentry** elements. (Each child element specifies the value of an element of the array property)

The following table describes the **field** XML element:

XML Element	field
Description	Specifies the value of a simple field value within a structure.
XML Attributes	• **name** (required) : the name of the field. • **value** (required) : the simple value of the field.
Contains	Has no children.

The following table describes the **structfield** XML element:

XML Element	structfield
Description	Specifies the value of a structure-typed field value within a structure.
XML Attributes	• **name** (required) : the name of the field.
Contains	One or more **field** and/or **structfield** and/or **arrayfield** elements. (Each child element specifies the value of a field of the sub-structure value)

The following table describes the **arrayfield** XML element:

XML Element	arrayfield
Description	Specifies the value of an array-typed field value within a structure.
XML Attributes	• **name** (required) : the name of the field.
Contains	Zero or more **entry** and/or **xyentry** and/or **structentry** and/or **arrayentry** elements. (Each child element specifies the value of an element of the array value. The children are given in the order in which they appear in the array)

The following table describes the **entry** XML element:

XML Element	entry
Description	Specifies the value of a simple entry within an array.
XML Attributes	• **value** (required) : the simple value of the array entry.
Contains	Has no children.

The following table describes the **xyentry** XML element:

XML Element	xyentry
Description	Specifies the value of an (x,y) coordinate-typed entry within an array.
XML Attributes	• **x** (required) : the x coordinate of the array entry. • **y** (required) : the y coordinate of the array entry.
Contains	Has no children.

The following table describes the **structentry** XML element:

XML Element	structentry
Description	Specifies the value of a structure-typed entry within an array.
XML Attributes	None
Contains	One or more **field** and/or **structfield** and/or **arrayfield** elements. (Each child element specifies the value of a field of the sub-structure value)

The following table describes the **arrayentry** XML element:

XML Element	arrayentry
Description	Specifies the value of an array-typed entry within an array. Note: this could be a way to specify multi-dimensional arrays.
XML Attributes	None
Contains	Zero or more **entry** and/or **xyentry** and/or **structentry** and/or **arrayentry** elements. (Each child element specifies the value of an element of the array value)

6.3 Document Type Definition (DTD) Specification

This section describes the grammar in DTD format. In practice, the actual DTD used may be a superset of this grammar, with OEM-specific XML elements added.

```
<!ELEMENT a661_df ( model, (symboltable)?, (picturetable)*, (a661_layer)+ )>

<!ATTLIST a661_df
    name CDATA #IMPLIED
    library_version CDATA #REQUIRED
    supp_version CDATA #REQUIRED>

<!-- === SYMBOL GRAPHICAL DEFINITION === -->

<!ELEMENT symboltable ( (symboldefn)* )>

<!ELEMENT symboldefn ( model, stdrepr, (focusrepr)?, (highlightrepr)?,
                       (rectangular_sensitive_area | circular_sensitive_area)? )>

<!ATTLIST symboldefn
    name CDATA #IMPLIED>

<!ELEMENT stdrepr ( (symboldefncmd)* )>

<!ELEMENT focusrepr ( (symboldefncmd)* )>

<!ELEMENT highlightrepr ( (symboldefncmd)* )>

<!ELEMENT rectangular_sensitive_area ( model )>

<!ELEMENT circular_sensitive_area ( model )>

<!ELEMENT symboldefncmd ( model )>

<!ATTLIST symboldefncmd
    name CDATA #IMPLIED
    type CDATA #REQUIRED>

<!-- === PICTURE DEFINITION === -->

<!ELEMENT picturetable ( (picturedefn)* )>

<!ELEMENT picturedefn ( model )>

<!ATTLIST picturedefn
    name CDATA #IMPLIED>

<!-- === LAYERS AND WIDGETS === -->

<!ELEMENT a661_layer ( model, (a661_widget)* )>

<!ATTLIST a661_layer
    name CDATA #IMPLIED>

<!ELEMENT a661_widget ( model, (a661_widget)* )>

<!ATTLIST a661_widget
    name CDATA #IMPLIED
    type CDATA #REQUIRED>
```

```
<!-- === PROPERTIES === -->

<!ELEMENT model ( (prop | structprop | arrayprop)* )>

<!-- property value: the "name" attribute gives the name of the property.

     prop ............ the value is simple.
     structprop ...... the value is a structure.
     arrayprop ....... the value is an array.

 -->

<!ELEMENT prop EMPTY>

<!ATTLIST prop
   name CDATA #REQUIRED
   value CDATA #REQUIRED>

<!ELEMENT structprop ( (field | structfield | arrayfield)+ )>

<!ATTLIST structprop
   name CDATA #REQUIRED>

<!ELEMENT arrayprop ( (entry | xyentry | arrayentry | structentry )* )>

<!ATTLIST arrayprop
   name CDATA #REQUIRED>


<!-- structure field values: the "name" attribute gives the name
     of the field.
     field ........ the value is simple.
     arrayfield ... the value is an array.
     structfield .. the value is a structure.

-->

<!ELEMENT field EMPTY >

<!ATTLIST field
    name CDATA #REQUIRED
    value CDATA #REQUIRED>

<!ELEMENT arrayfield ( (entry | xyentry | structentry | arrayentry)* )>

<!ATTLIST arrayfield
    name CDATA #REQUIRED>

<!ELEMENT structfield ( (field | arrayfield | structfield)+ )>

<!ATTLIST structfield
    name CDATA #REQUIRED>


<!-- array entries:

     entry ........ the array entry is simple.
     xyentry ...... the array entry is an (x,y) pair.
     arrayentry ... the array entry is an array.
     structentry .. the array entry is a structure.

-->

<!ELEMENT entry EMPTY >

<!ATTLIST entry
    value CDATA #REQUIRED>

<!ELEMENT xyentry EMPTY >

<!ATTLIST xyentry
    x CDATA #REQUIRED
    y CDATA #REQUIRED>

<!ELEMENT arrayentry ( (entry | xyentry | arrayentry | structentry )* )>
```

```
<!ELEMENT structentry ( (field | arrayfield | structfield)+ )>
```

6.4 References

- “eXtensible Markup Language (XML)”, a technical recommendation standard of the W3C. W3C Consortium (www.w3c.org) , release 1.0, February 10, 1998
- “XML Schema”, a technical recommendation standard of the W3C Consortium (www.w3c.org) , release 1.0, May 2, 2001.

7.1 Introduction

This section defines how bitmap images can be defined in Definition Files. These bitmap images are identified by a PictureReference number, and they can be referenced by a variety of widgets, such as Picture and PicturePushButton.

Similar to vector symbols defined by the Symbol Graphical Definition language (see section 5 of this specification), bitmaps defined in a Definition File can be referenced by any layer whose User Application Layer Definition (UALD) is defined in the same Definition File as the symbol it wants to reference.

The following scheme should be used to find the appropriate picture for a given PictureReference number:

- **If a Picture with matching ID is found in the DF, use it**
- Otherwise, use the predefined (global) Picture with that ID

7.2 Picture Definition Structures

Pictures are defined in a Picture structure, which in turn is part of a PictureBlock structure. Each PictureBlock determines the pixel format and color table (if applicable) for all Pictures that it contains.

Table 7-1 below defines the PictureBlock structure. Zero or more PictureBlock structures may appear inside a Definition File between the Definition File header and the User Application Layer Definitions (UALDs), as described in Section 4.5.3.2.

Table 7-1 – PictureBlockStructure

A661_Picture_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_PICTURE_BLOCK	uchar	8	Start keyword opening a picture.
PixelFormat	uchar	8	Defines Pixel and storage format, e.g. A661_PIX_FMT_RGBA_8
NumberOfPicturesInBlock	uchar	8	Number of pictures defined in this block
NumberOfColorTableEntries	uchar	8	Size (number of colors) of the color table. 0 means global CDS color table is referenced, only used with PixelFormats that reference a color table
ColorTableFormat	uchar	8	Defines format of each color table entry, for example A661_PIX_FMT_RGBA_8. Ignored if NumberOfColorTableEntries is zero, only used with PixelFormats that reference a color table.
UnusedPad	ushort	24	
[Color_Table]		{32}	Color table entries (if defined), padded to align with 4 byte boundary, only used with PixelFormats that reference a color table
{ A661_Picture_Structure_DT }+	N/A	{32}+	One or more pictures in this block
Reserved for future use	N/A	32	
A661_END_PICTURE_BLOCK	uchar	8	Keyword ending a block of symbol information.
UnusedPad	N/A	24	0

Pictures within these PictureBlocks are defined as described in Table 7-2.

Table 7-2 – PictureStructure

A661_Picture_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_PICTURE	uchar	8	Start keyword opening a picture.
UnusedPad	N/A	8	
PictureReference	ushort	16	
NumberOfPixelsWidth	ushort	16	Width of the picture in pixels
NumberOfPixelsHeight	ushort	16	Height of the picture in pixels
{ A661_Pixel_Values }+	N/A	{32}+	Information for each pixel, padded to align with 4 byte boundary after the last pixel
A661_END_PICTURE	uchar	8	Keyword ending a block of symbol information.
UnusedPad	N/A	24	0

In this structure, PictureReference is a numeric identifier that widgets such as Picture and PicturePushButton use to identify a bitmap they wish to display. Pictures are rectangular; the size of a picture is defined by the width and height, measured in the number of pixels used in the bitmap that defines the picture. Note that the details of scaling or mapping this image to pixels of the actual display is dependent on the CDS implementation.

Pixels data ({ A661_Pixel_Values }+) is stored as a sequence of pixels, starting with the top left pixel and then progressing left to right, one line at a time until the bottom line is completed.

The definition of pixel and storage formats is addressed by an enumerated type, which determines both the pixel format and the storage format. While the support for various pixel and storage formats is largely dependent on the CDS implementation, three widely used formats are defined in this specification. CDS implementers may adopt these formats at their discretion or add additional formats as required.

Table 7-3 – Pixel and Storage Formats

Enumeration Type Value	Pixel Format	Storage Format (bits)
A661_PIX_FMT_RGBA_8	Red, Green, Blue, Alpha	8+8+8+8=32
A661_PIX_FMT_LUMINANCE_ALPHA_8	Luminance, Alpha	8+8=16
A661_PIX_FMT_COLOR_INDEXED_8	Color Index	8

Red, Green, and Blue information in the formats shown in Table 7-3 are stored in 8 bits each, where zero means lowest intensity and 255 means highest intensity of each base color. For Alpha, zero means fully transparent and 255

means fully opaque. Likewise, a luminance of zero has the lowest intensity, and 255 represents the highest intensity.

7.3 Color Tables

Pictures may reference a color table, in which case indices into this table are used instead of explicit RBG values. There are two different scenarios:

1. NumberOfColorTableEntries = 0:

   No color table is defined in the PictureBlock. Instead, color index values in the picture are interpreted the same way as ColorIndex parameters in other widgets, such as GpLine. This means that the global CDS color table is used for the picture. The ColorTableFormat parameter is ignored.

2. NumberOfColorTableEntries > 0:

   In this case, the NumberOfColorTableEntries parameter defines how many colors indices are available. The range of color index values is from zero to (NumberOfColorTableEntries – 1), meaning that at most a PictureBlock can define 255 colors in a color table. The ColorTableFormat parameter determines the format of the data describing each color table entry, starting with color index zero.

APPENDIX A
GLOSSARY

Active Layer
When a layer is active, the CDS updates widget parameters of this layer (refer to Section 2.3.2.4 – Layer Activity/Visibility Management).

ARINC 661 Widget Library
Widget Library resident in the CDS containing the full description (graphic and behavior) of each ARINC 661 widget that a user application may require for displaying.

Cockpit Display System (CDS)
Equipment that performs the functions described in this document.

Crew member interaction (or crew member input)
Action of a crew member on an interactive widget through the use of an input device.

Cursor
Visual indicator of the point of crew input on the screen.

Definition File (DF)
A Definition File is associated with one UA. The DF contains the UA Layer Description (UALD) to be displayed on the CDS.

Definition phase
Consists in loading of DF in the CDS, which specify the creation of widgets in order to describe user application's interface layouts. The instantiation (creation plus first setting of all parameters) of widgets inside the CDS is part of the definition phase. This will occur before the beginning of the run-time phase.

Disabled
State of an interactive widget when it does not react to crew-member activation.

Display
A non-specific term, generally meaning "thing you look at." As a noun, "display" refers to a physical assembly of metal and glass (Display Head), or to the pattern of colored dots that appear on that glass (Format). As a verb, "display" refers to the act of deciding which of those colored dots to light (render) or, loosely, to providing the parameters necessary to be able to render.

Display Head
A physical assembly that can generate patterns of colored dots (raster) or lines (stroke).

Event
Notification sent by the CDS to a UA to indicate a crew member interaction has occurred on a widget owned by that UA.

Focus
State of a widget in which this widget receives the events triggered by a crew member through a keyboard or other device, such as rotary wheels, except for the cursor control device.

APPENDIX A
GLOSSARY

Format

Format image rendered to the whole display unit surface. A format is constructed from one or more windows

Highlight

A widget is highlighted when the cursor over-flies its interactive area. A click with the cursor control device on an highlighted widget will bring the focus on this widget (refer to Focus). Depending on the OEM choice, the click may also select the widget.

Inner state

Specific states of a widget. This state level represents the core of the widget behavior as well as its functional objectives. Examples of inner states :

> For a basic PushButton, there is one inner state.
> For a CheckButton, there are two inner states, which are ‘SELECTED’ and ‘UNSELECTED’

Interactive

Category of widgets that can generate events in response to crew member activity on input device(s).

Layer

Layers provide the mechanism to combine graphical information from several UAs inside one window. For example, layers of the ND image include compass rose, map with interactive way-points, TCAS information, terrain or weather display. A layer is connected to a unique UA, whereas a UA can use several layers.

A layer is the higher level entity of the CDS that is known by the UA. From the UA point of view, the Layer is the high level container in the hierarchical structure of the UA widgets. From the CDS point of view, the layer is one graphical layer associated with one UA inside a window. The layer layout within a window is beyond the scope of this standard. Refer to Section 2.3.2, Layer Definition.

Look & Feel

Cover the graphical characteristics of a widget, which are not managed by a UA but by the CDS in order to insure a homogeneous HMI. This terminology also applies to widget behavior internal to the CDS, leading to a state diagram internal to the CDS that manages transition between visual representations.

Mask

A graphical representation (a picture, typically a bitmap) used to implement non-rectangular clipping. The exact format is CDS-specific. Typically, a Mask is a Picture made up of only Black and Transparent elements.

Navigation Display (ND)

Generally, the ND includes the Course/Speed/Flight Plan/Surveillance indicators.

Normal

Normal visual representation of the widget when it is visible, enabled, and not selected.

Picture

A fixed image, stored in the CDS, referenced by an index, not rotatable.

APPENDIX A
GLOSSARY

Race condition

Race condition occurs when there is a cross of messages between the CDS and a UA concerning dynamic widgets. It can lead to some inconsistency between the UA context and the CDS display.

Reference to …

The index or ID of something that has been loaded into the CDS.

Render

The act of combining software-code instructions, uploaded symbols and parameters into a pattern of colored dots or lines on a display head.

Run-time phase

The run-time phase consists of dynamic data transfers between UAs and CDS using ARINC 661 run-time commands.

Style guide

The style guide defined by the OEM should describe the Look & Feel (common graphical characteristics and consistent behavior) inside the cockpit, and thus, provide necessary information to UAs for their HMI interface design.

Symbol

A rotatable image, stored in the CDS, referenced by an index.

User Application Layer Definition (UALD)

The UALD describes the structure of data of one layer of a UA. This set of data describes all the widgets that have to be allocated in the CDS inside this layer and describes widget parameters values as well as the widget tree to be drawn.

User Application (UA)

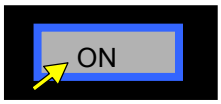
A UA can use several layers in one window to draw its graphical objects, typically to insure graphical priorities among layers.

Visual representation

A widget is characterized by its inner states. One inner state covers several graphical representations. The visual representation “state” is managed internal to the CDS.

An **example** of visual representations for a Button follows:

Highlighted:



Normal:

Widget

Graphical-user interface, object between a crew member and the UAs. Widgets belong to a Widget Library inside the CDS. A widget may (or not) be interactive (i.e.

accept and react to crew member actions). A widget is defined by a set of characteristics accessible to UA through ARINC 661 parameters, some functional states corresponding to specific sets of graphical parameters , which refers to “Look & Feel,” and a behavior.

Window

A window defines a rectangular physical area of the display surface. A window consists of one or more layers and is controlled by the CDS.

APPENDIX B
ACRONYMS AND ABBREVIATIONS

ADI	Attitude Director Indicator
AEEC	Airlines Electronic Engineering Committee
BITE	Built-In Test Equipment
CCD	Cursor Control Device
CDS	Cockpit Display System
CPU	Central Processing Unit
DCDU	Data-link Control Display Unit
DF	Definition File
DU	Display Unit
EFI	Electronic Flight Instrument
EFIS	Electronic Flight Instrumentation System
EICAS	Engine Indication and Crew Alerting System
FM	Flight Management
FMS	Flight Management System
GNLU	GNSS Navigation and Landing Unit
GNU	GNSS Navigation Unit
HMI	Human Machine Interface
HUD	Head-Up Display
IRS	Inertial Reference System
I/O	Input/Output
L1 ,L2 ,L3	Layer 1, Layer 2, Layer 3
LRU	Line Replacement Unit
LSB	Least Significant Bit
MCDU	Multi-Purpose Display Unit
MFD	Multi-Function Display
MSB	Most Significant Bit

APPENDIX B
ACRONYMS AND ABBREVIATIONS

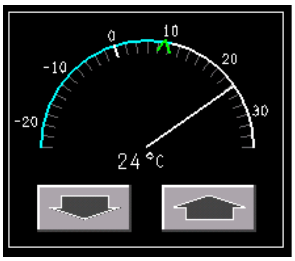
ND	Navigation Display
OEM	Original Equipment Manufacturer
PFD	Primary Flight Display
PRP	Projection Reference Point
TAWS	Terrain Avoidance Warning System
UA	User Application
UALD	User Application Layer Definition
WXR	Weather Radar
2D	Two Dimensional
3D	Three Dimensional

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-1 User Application Example Overview

The following UA example controls the cabin temperature using two interfaces:

1. The UA is connected to the aircraft environment with:
   a. one input: a cabin temperature sensor
   b. one output: an actuator for the heater system and cooler system
2. The UA is connected to the CDS with a ARINC 661 interface to display the following format in a DU window:



Buttons increment or decrement a selected temperature for the cabin, indicated by the small green pointer. The current cabin temperature is indicated by both the white arrow and the digital readout.

The format is composed of eight ARINC 661 widgets summarized as follows:

The scale is an ARINC 661 picture (A661_PICTURE : several colors, no rotation).

Pointers (selected and real temperature) are ARINC 661 symbols (A661_SYMBOL: can rotate, has one selected color).

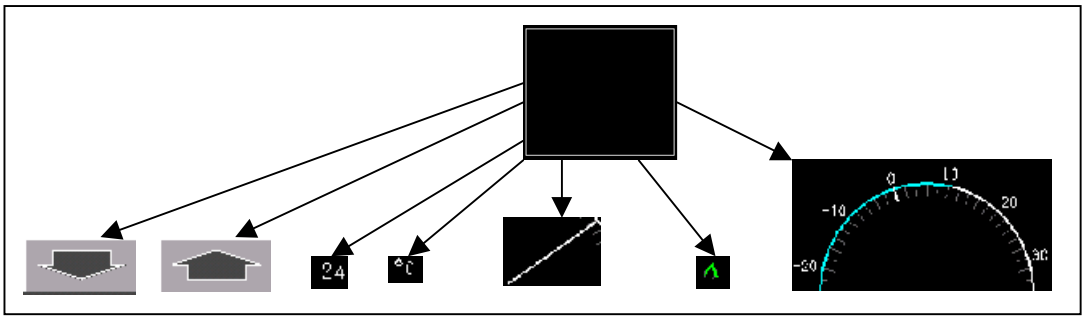
The digital readout and its unit string are ARINC 661 labels (A661_LABEL).

Buttons are ARINC 661 buttons using a reference to a symbol (A661_PICTURE_PUSH_BUTTON).

All drawings are clipped inside an ARINC 661 panel container (A661_PANEL).

APPENDIX C
EXAMPLE OF A DEFINITION FILE

Each widget is a node in a hierarchical structure defined in the DF. The tree for Example C.1 is:



C-2 Example of Application Code at Run Time

An example of UA code **for controlling the cabin temperature example** at run time follows:

```
if ((selectedCabinTemp < maxValue) and (BUTTON_PRESSED.id = IncreaseSelectTemp)) then
        increment(selectedCabinTemp);

end if
if ((selectedCabinTemp > minValue) and (BUTTON_PRESSED.id = DecreaseSelectTemp)) then
        decrease(selectedCabinTemp);

end if
Actuator.heaterCommand = PIDcontroller (selectedCabinTemp);
angleValue = selectedCabinTemp * scaleFactor + offset;
setParameter(TemperatureSelectedPointer, RotationAngle, angleValue);
        cockpitTemp = Sensor.cabinTemp
if IsValid(cockpitTemp) = True then
        angleValue = cockpitTemp * scaleFactor + offset;

        setParameter(IndicatedTempDRO, LabelString, toString(cockpitTemp));
        setParameter(TemperatureIndicatedPointer, RotationAngle, angleValue);
        if (cockpitTemp > ThresholdValue ) then
                setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_WARNING);
                setParameter(TemperatureIndicatedPointer, StyleSet, A661_STYLE_SET_WARNING);
else
                setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_NOMINAL);
                setParameter(TemperatureIndicatedPointer, StyleSet, A661_STYLE_SET_NOMINAL);

        end if
setParameter(TemperatureIndicatedPointer, Visible, A661_TRUE);
        setParameter(IncreaseSelectTemp, Enable, A661_TRUE);
        setParameter(DecreaseSelectTemp, Enable, A661_TRUE);
else
        setParameter(IndicatedTempDRO, LabelString, "---");
        setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_WARNING);
        setParameter(TemperatureIndicatedPointer, Visible, A661_FALSE);
        beginBlock();
        setParameter(IncreaseSelectTemp, Enable, A661_FALSE);
        setParameter(DecreaseSelectTemp, Enable, A661_FALSE);
        endBlock();
end if
```

Begin-Block and End-Block commands limit the amount of data that can be processed as coherent information. In the following example, the corresponding byte stream is sent for this block by the UA assuming that the network allows transmission of blocks of such size. In other cases, sub-blocks might be used.

APPENDIX C
EXAMPLE OF A DEFINITION FILE

Paragraphs are aligned with 32 bits length words.

```
                        # Word 1
B0                      # A661_BEGIN_BLOCK
42                      # LAYER ID
1230                    # CONTEXT NUMBER

                        # Word 2
00000024                # BLOCK SIZE (= 36 bytes, words 1 - 9)

                        # Word 3
CA02                    # A661_CMD_SET_PARAMETER
000C                    # COMMAND SIZE (= 12 bytes, words 3 - 5)

                        # Word 4
0000                    # UNUSED PAD
4566                    # WIDGET ID (IncreaseSelectTemp)

                        # Word 5
B180                    # PARAMETER ID (A661_ENABLE)
0000                    # VALUE (A661_FALSE)

                        # Word 6
CA02                    # A661_CMD_SET_PARAMETER
000C                    # COMMAND SIZE (= 12 bytes, words 6 - 8)

                        # Word 7
0000                    # UNUSED PAD
4567                    # WIDGET ID (DecreaseSelectTemp)

                        # Word 8
B180                    # PARAMETER ID (A661_ENABLE)
0000                    # VALUE (A66_FALSE)

                        # Word 9
D0                      # A661_END_BLOCK
000000                  # UNUSED PAD
```

When a CCD click occurs on one of the buttons, the following message is sent from the CDS to the UA:

```
                        # Word 1
B0                      # A661_BEGIN_BLOCK
42                      # LAYER ID
1230                    # CONTEXT NUMBER

                        # Word 2
00000018                # BLOCK SIZE (= 24 bytes, words 1 - 6)

                        # Word 3
CC01                    # A661_NOTIFY_WIDGET_EVENT
000C                    # COMMAND SIZE (= 12 bytes, words 3 - 5)

                        # Word 4
4566                    # WIDGET ID (IncreaseSelectTemp)

CCD1                    # EVENT ORIGIN


                        # Word 5
E060                    # EVENT ID (A661_SELECTION)
0000                    # UNUSED PAD

                        # Word 6
D0                      # A661_END_BLOCK
000000                  # UNUSED PAD
```

The UA then acknowledges the event by sending an acknowledgment back to the CDS.

C-3 Definition File

Section C.3 provides an example of a User Application Definition File (UADF) for the cabin temperature UA, containing only one layer. Note: unit length of measure is 1/100 of millimeter.

UADF Example for Cabin Temperature Application

```
# START DF
# Input file:
cabin_temperature.xml
# Hexadecimal               # Comment

                            # Word 1
A661                        # A661_DF_MAGIC_NUMBER
00                          # A661 LIBRARY VERSION
02                          # A661 SUPP VERSION

                            # Word 2
6788                        # DF ID
0000                        # SIZE OF OEM FREE DATA

                            # Word 3
A0                          # A661_BEGIN_LAYER_BLOCK
42                          # LAYER ID
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
1230                        # CONTEXT NUMBER

                            # Word 4
0000013C                    # BLOCK SIZE (=316 bytes, words 3 - 81)

                            # Widget instance number:1
                            # Word 5
CA01                        # A661_CMD_CREATE
0020                        # COMMAND SIZE (= 32 bytes, words 5 - 12)
A1F0                        # WIDGET TYPE (A661_PANEL)
1221                        # WIDGET ID (CabinTempPanel)
0000                        # PARENT ID (zero indicates layer is parent)
01                          # Enable, value:A661_TRUE
01                          # Visible, value:A661_TRUE
00002AF8                    # PosX, value: 11000 = 110 mm, 4.33 in
0000319C                    # PosY, value: 12700 = 127 mm, 5 in
00001DC4                    # SizeX, value: 7620 = 76.2 mm, 3 in
0000175A                    # SizeY, value: 5978 = 59.78 mm, 2.44 in
0000                        # StyleSet, value:A661_STYLE_SET_DEFAULT
00                          # MotionAllowed, value: A661_FALSE
00                          # UNUSED PAD

                            # Widget instance number:2
                            # Word 13
CA01                        # A661_CMD_CREATE
0020                        # COMMAND SIZE (=32 bytes, words 13 - 20)
A200                        # WIDGET TYPE (A661_PICTURE)
1222                        # WIDGET ID (TemperatureCelciusScale)
1221                        # PARENT ID (CabinTempPanel)
00                          # Anonymous, value:A661_FALSE
01                          # Visible, value:A661_TRUE
000002EE                    # PosX, value: 750, 7.5 mm, 0.31 in
00000956                    # PosY, value: 2390, 23.9 mm, 0.98 in
000017EE                    # SizeX, value: 6126, 61.26 mm, 2.41 in
000009BA                    # SizeY, value: 2490, 24.9 mm, 1.02 in
0000                        # StyleSet, value: A661_STYLE_SET_DEFAULT
9870                        # PictureRef

                            # Widget instance number:3
                            # Word 21
CA01                        # A661_CMD_CREATE
0020                        # COMMAND SIZE (=32 bytes, words 21 - 28)
A310                        # WIDGET TYPE (A661_SYMBOL)
1223                        # WIDGET ID (TemperatureIndicatedPointer)
1221                        # PARENT ID (CabinTempPanel)
00                          # MotionAllowed, value: A661_FALSE
01                          # Visible, value:A661_TRUE
00000EE2                    # PosX, value:3810, 38.1 mm, 1.56 in
00000956                    # PosY, value: 2390, 23.9 mm, 0.98 in
0000238C                    # RotationAngle, value: 9100 = 50 deg [fr(180) LSB =
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
                        0.0005439]
0001                    # StyleSet, value: OEM_STYLESET_FREE_COLOR
9874                    # PictureReference,
                        value:SymbolTemperatureIndicatedPointer
0F                      # ColorIndex, value: OEM_WHITE
000000                  # UNUSED PAD


                         # Widget instance number:4
                         # Word 29
CA01                     # A661_CMD_CREATE
0020                     # COMMAND SIZE (=32 bytes, words 29 - 36)
A310                     # WIDGET TYPE (A661_SYMBOL)
1224                     # WIDGET ID (TemperatureSelectedPointer)
1221                     # PARENT ID (CabinTempPanel)
00                       # MotionAllowed, value: A661_FALSE
01                       # Visible, value:A661_TRUE
00000EE2                 # PosX, value:3810, 38.1 mm, 1.56 in
00000956                 # PosY, value: 2390, 23.9 mm, 0.98 in
0000071C                 # RotationAngle, value: 1820 = 10 deg [fr(180) LSB =
                         0.0005439]
0001                     # StyleSet, value: OEM_STYLESET_FREE_COLOR
0003                     # SymbolReference,
                         value:SymbolTemperatureSelectedPointer
01                       # ColorIndex, value: OEM_GREEN4
000000                   # UNUSED PAD


                        # Widget instance number:5
                        # Word 37
CA01                    # A661_CMD_CREATE
2C                      # COMMAND SIZE (=44 bytes, words 37 - 47)
A160                    # WIDGET TYPE (A661_LABEL)
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
1225                    # WIDGET ID (IndicatedTempDRO)
1221                    # PARENT ID (CabinTempPanel)
00                      # Anonymous, value:A661_FALSE
01                      # Visible, value:A661_TRUE
00000B30                # PosX, value: 2864, 28.64 mm, 1.13 in
000006E8                # PosY, value: 1768, 17.68 mm, 0.72 in
000003B6                # SizeX, value: 950, 9.5 mm, 0.39 in
0000026E                # SizeY, value: 622, 6.22 mm, 0.25 in
00000000                # RotationAngle, value: 0 deg [fr(180) LSB =
                        0.0005439]
0801                    # StyleSet, value:OEM_STYLESET_NORMAL_READOUT
0004                    # MaxStringLength, value:4
00                      # MotionAllowed, value:A661_FALSE
00                      # Font, value: OEM_STYLESET_DEFAULT_FONT
00                      # UNUSED PAD
14                      # Alignment, value:A661_RIGHT
32340000                # LabelString, value: "24"

                        # Widget instance number:6
                        # Word 48
CA01                    # A661_CMD_CREATE
2C                      # COMMAND SIZE (=44 bytes, words 48 - 58)
A160                    # WIDGET TYPE (A661_LABEL)
1226                    # WIDGET ID (IndicatedUnitLabel)
1221                    # PARENT ID (CabinTempPanel)
00                      # Anonymous, value:A661_FALSE
01                      # Visible, value:A661_TRUE
00000B30                # PosX, value: 3984, 39.84 mm, 1.63 in
000006E8                # PosY, value: 1768, 17.68 mm, 0.72 in
000001D9                # SizeX, value: 473, 4.73 mm, 0.19 in
0000026E                # SizeY, value: 622, 6.22 mm, 0.25 in
00000000                # RotationAngle, value: 0 deg [fr(180) LSB =
                        0.0005439]
0801                    # StyleSet, value:OEM_STYLESET_NORMAL_READOUT
0002                    # MaxStringLength, value:2
00                      # MotionAllowed, value:A661_FALSE
00                      # Font, value: OEM_STYLESET_DEFAULT_FONT
00                      # UNUSED PAD
13                      # Alignment, value:A661_LEFT
81430000                # LabelString, value: "°C"

                        # Widget instance number:7
                        # Word 59
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
CA01                    # A661_CMD_CREATE
0020                    # COMMAND SIZE (=32 bytes, words 62 - 69)
A240                    # WIDGET TYPE (A661_PICTURE_PUSH_BUTTON)
1227                    # WIDGET ID (IncreaseSelectTemp)
1221                    # PARENT ID (CabinTempPanel)
01                      # Enable, value:A661_TRUE
01                      # Visible, value:A661_TRUE
000001D9                # PosX, value: 473, 4.73 mm, 0.19 in
000001D9                # PosY, value: 473, 4.73 mm, 0.19 in
00000B30                # SizeX, value: 2864, 28.64 mm, 1.13 in
000003B6                # SizeY, value: 950, 9.5 mm, 0.39 in
0000                    # StyleSet, value:A661_STYLE_SET_DEFAULT
0000                    # NextFocusedWidget, value:0
9878                    # PictureReference, value:SymbolArrowUp
0000                    # MaxStringLength, value:0
15                      # PicturePosition, value:A661_CENTER
00                      # AutomaticFocusMotion, value:A661_FALSE
00                      # Alignment, value: A661_CENTER
00                      # UNUSED PAD
00000000                # LabelString, value: ""

                        # Widget instance number:8
                        # Word 70
CA01                    # A661_CMD_CREATE
002C                    # COMMAND SIZE (=44 bytes, words 70 - 80)
A240                    # WIDGET TYPE (A661_PICTURE_PUSH_BUTTON)
1228                    # WIDGET ID (DecreaseSelectTemp)
1221                    # PARENT ID (CabinTempPanel)
01                      # Enable, value:A661_TRUE
01                      # Visible, value:A661_TRUE
00000B30                # PosX, value: 3984, 39.84 mm, 1.63 in
000001D9                # PosY, value: 473, 4.73 mm, 0.19 in
00000B30                # SizeX, value: 2864, 28.64 mm, 1.13 in
000003B6                # SizeY, value: 950, 9.5 mm, 0.39 in
0000                    # StyleSet, value:A661_STYLE_SET_DEFAULT
0000                    # NextFocusedWidget, value:0
987C                    # PictureReference, value:SymbolArrowDown
0000                    # MaxStringLength, value:0
15                      # PicturePosition, value:A661_CENTER
00                      # AutomaticFocusMotion, value:A661_FALSE
00                      # Alignment, value: A661_CENTER
00                      # UNUSED PAD
00000000                # LabelString, value: ""
                        # Word 81
C0                      # A661_END_LAYER_BLOCK
000000                  # UNUSED PAD
E0                      # A661_DF_FOOTER
000000                  # UNUSED PAD
# END DF
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-4 Example of the XML Form of the Definition File

For the application illustrated in Section C.3, the corresponding XML form of the DF is as follows. Refer to Section 6 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values.

Note: the DOCTYPE specification is ad hoc. It requires that a file defining the DTD be found in the current directory with the name “a661.dtd”.

```
<?xml version="1.0"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="0" supp_version="2">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <a661_layer>
      <model>
         <prop name="LayerId" value="66"/>
         <prop name="ContextNumber" value="0x1230"/>
         <prop name="Height" value="20000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="CabinTempPanel" type="A661_PANEL">
         <model>
            <prop name="WidgetId" value="4641"/>
            <prop name="PosX" value="11000"/>
            <prop name="PosY" value="12700"/>
            <prop name="SizeX" value="7620"/>
            <prop name="SizeY" value="5978"/>
         </model>
         <a661_widget name="TemperatureCelciusScale" type="A661_PICTURE">
            <model>
               <prop name="WidgetId" value="4642"/>
               <prop name="PosX" value="750"/>
               <prop name="PosY" value="2390"/>
               <prop name="SizeX" value="6126"/>
               <prop name="SizeY" value="2490"/>
               <prop name="PictureReference" value="SymbolTemperatureCelciusScale"/>
            </model>
         </a661_widget>
         <a661_widget name="TemperatureIndicatedPointer" type="A661_SYMBOL">
            <model>
               <prop name="WidgetId" value="4643"/>
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="9100" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_WHITE" />
               <prop name="PictureReference" value="SymbolTemperatureIndicatedPointer" />
            </model>
         </a661_widget>
         <a661_widget name="TemperatureSelectedPointer" type="A661_SYMBOL">
            <model>
               <prop name="WidgetId" value="4644"/>
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="1820" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_GREEN4" />
               <prop name="PictureReference" value="SymbolTemperatureSelectedPointer" />
            </model>
         </a661_widget>
         <a661_widget name="IndicatedTempDRO" type="A661_LABEL">
            <model>
               <prop name="WidgetId" value="4645"/>
               <prop name="PosX" value="2864" />
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                <prop name="PosY" value="1768" />
                <prop name="SizeX" value="950" />
                <prop name="SizeY" value="622" />
                <prop name="StyleSet" value="OEM_STYLESET_NORMAL_READOUT" />
                <prop name="Font" value="OEM_STYLESET_DEFAULT_FONT" />
                <prop name="MaxStringLength" value="4" />
                <prop name="Alignment" value="A661_RIGHT" />
                <prop name="LabelString" value="24" />
             </model>
          </a661_widget>
          <a661_widget name="IndicatedUnitLabel" type="A661_LABEL">
             <model>
                <prop name="WidgetId" value="4645"/>
                <prop name="PosX" value="3984" />
                <prop name="PosY" value="1768" />
                <prop name="SizeX" value="473" />
                <prop name="SizeY" value="622" />
                <prop name="StyleSet" value="OEM_STYLESET_NORMAL_READOUT" />
                <prop name="Font" value="OEM_STYLESET_DEFAULT_FONT" />
                <prop name="MaxStringLength" value="2" />
                <prop name="LabelString" value="°C" />
             </model>
          </a661_widget>
          <a661_widget name="IncreaseSelectTemp" type="A661_PICTURE_PUSH_BUTTON">
             <model>
                <prop name="WidgetId" value="4646"/>
                <prop name="PosX" value="473" />
                <prop name="PosY" value="473" />
                <prop name="SizeX" value="2864" />
                <prop name="SizeY" value="950" />
                <prop name="PictureReference" value="SymbolArrowUp" />
             </model>
          </a661_widget>
          <a661_widget name="DecreaseSelectTemp" type="A661_PICTURE_PUSH_BUTTON">
             <model>
                <prop name="WidgetId" value="4647"/>
                <prop name="PosX" value="3984" />
                <prop name="PosY" value="473" />
                <prop name="SizeX" value="2864" />
                <prop name="SizeY" value="950" />
                <prop name="PictureReference" value="SymbolArrowDown" />
             </model>
          </a661_widget>
       </a661_widget>
    </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-5 A More Interesting XML DF Example

This is an XML file example that uses a wider range of property types. Refer to Section 6 for a description of the XML DF format. **In this example not all property values are given explicitly. The unspecified properties take default values.**

Note: the DOCTYPE specification is ad hoc: it requires that a file defining the DTD be found in the current directory with the name “a661.dtd”.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="3" supp_version="2">
   <model>
      <prop name="ApplicationId" value="45"/>
   </model>
   <a661_layer name="ATC_COM">
      <model>
         <prop name="LayerId" value="55"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="10000"/>
         <prop name="Width" value="10000"/>
      </model>
      <a661_widget name="PNL_COM" type="A661_PANEL">
         <model>
            <prop name="WidgetId" value="1"/>
            <prop name="Visible" value="A661_TRUE"/>
            <prop name="Enable" value="A661_TRUE"/>
            <prop name="StyleSet" value="0"/>
            <prop name="PosX" value="100"/>
            <prop name="PosY" value="100"/>
            <prop name="SizeX" value="7000"/>
            <prop name="SizeY" value="5000"/>
         </model>
         <a661_widget name="CKB_CHOICE" type="A661_CHECK_BUTTON">
            <model>
               <prop name="WidgetId" value="2"/>
               <prop name="Visible" value="A661_TRUE"/>
               <prop name="Enable" value="A661_TRUE"/>
               <prop name="CheckButtonState" value="A661_UNSELECTED"/>
               <prop name="StyleSet" value="1"/>
               <prop name="PosX" value="100"/>
               <prop name="PosY" value="100"/>
               <prop name="SizeX" value="3000"/>
               <prop name="SizeY" value="900"/>
               <prop name="NextFocusedWidget" value="0"/>
               <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
               <prop name="LabelSring" value="SELECT"/>
               <prop name="MaxStringLength" value="7"/>
               <prop name="Alignment" value="A661_LEFT"/>
               <prop name="PicturePosition" value="A661_LEFT"/>
            </model>
         </a661_widget>
         <a661_widget name="SL_COUNTRY" type="A661_SCROLL_LIST">
            <model>
               <prop name="WidgetId" value="3"/>
               <prop name="Visible" value="A661_TRUE"/>
               <prop name="Enable" value="A661_FALSE"/>
               <prop name="StyleSet" value="2"/>
               <prop name="PosX" value="100"/>
               <prop name="PosY" value="1100"/>
               <prop name="SizeX" value="6800"/>
               <prop name="SizeY" value="3800"/>
               <prop name="NextFocusedWidget" value="0"/>
               <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
               <prop name="NumberOfEntries" value="4"/>
               <prop name="MaxNumberOfEntries" value="15"/>
               <prop name="FirstVisibleEntry" value="3"/>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                <prop name="FirstAccessibleEntry" value="3"/>
                <prop name="FlagReportVisibleEntry" value="A661_FALSE"/>
                <prop name="SelectedEntry" value="3"/>

                <structprop name="DefaultStyleText">
                   <field name="TOutline" value="0"/>
                   <field name="TBackColor" value="0"/>
                   <field name="TForeColor" value="41"/>
                   <field name="TFont" value="0"/>
                </structprop>
                <prop name="MaxStringLength" value="10"/>
                <prop name="Alignment" value="A661_CENTER"/>
                <arrayprop name="LabelStringArray">
                   <entry value="CANADA"/>
                   <entry value="USA"/>
                   <entry value="BELGIUM"/>
                   <entry value="FRANCE"/>
                   <entry value="GERMANY"/>
                   <entry value="UK"/>
                </arrayprop>
                <arrayprop name="EnableArray">
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                </arrayprop>
                <prop name="VerticalScroll" value="A661_RIGHT"/>
             </model>
          </a661_widget>
       </a661_widget>
       <a661_widget name="BF_COM" type="A661_BUFFER_FORMAT">
          <model>
             <prop name="WidgetId" value="4"/>
             <prop name="NumberOfFields" value="2"/>
             <arrayprop name="BufferStructure">
                <structentry>
                   <field name="WidgetIdent" value="2"/>
                   <field name="ParameterIdent" value="A661_INNER_STATE_CHECK"/>
                </structentry>
                <structentry>
                   <field name="WidgetIdent" value="3"/>
                   <field name="ParameterIdent" value="A661_ENABLE"/>
                </structentry>
             </arrayprop>
          </model>
       </a661_widget>
    </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-6 Binary DF Specifying Symbol Graphical Definitions

This section shows a binary Definition File that defines two symbols. See Section C.7 for an XML version of this example.

```
# Binary Definition File
# Example for ARINC-661 Specification
# Demonstrates how symbols and layers can be defined in
# a single binary definition file

A661            # A661_DF_MAGIC_NUMBER
02              # Library_Version
02              # Supp_Version
6788            # Application Identifier
0000            # UNUSED PAD

# Symbol Definition starts here:
F0              # A661_BEGIN_SYMBOL_BLOCK
000000          # UNUSED PAD
00000076        # block size (118 bytes)
CA04            # A661_CMD_CREATE_SYMBOL
0030            # command size (48 bytes)
0064            # symbol ID (100)
0000            # UNUSED PAD
90D0            # A661_SYMBOL_DEFN_POLYLINE
0004            # number of vertices
01              # closed (A661_TRUE)
000000          # UNUSED PAD
FFFFFE0C        # first X (-500)
FFFFFE0C        # first Y (-500)
000001F4        # second X (500)
000001F4        # second Y (500)
FFFFFE0C        # third X (-500)
000001F4        # third Y  (500)
000001F4        # fourth X (500)
FFFFFE0C        # fourth Y(-500)
CA04            # A661_CMD_CREATE_SYMBOL
003A            # command size (58 bytes)
0065            # symbol ID (101)
0000            # UNUSED PAD
90B0            # A661_SYMBOL_DEFN_LINE
0000            # UNUSED PAD
00000000        # first X (0)
FFFFFC18        # first Y (-1000)
00000000        # second X (0)
FFFFFE0C        # second Y (-500)
90F0            # A661_SYMBOL_DEFN_TRIANGLE
0000            # UNUSED PAD
00              # filled (A661_FALSE)
00              # UNUSED PAD
FFFFFF38        # first X (-200)
FFFFFE0C        # first Y (-500)
00000000        # second X (0)
00000000        # second Y (0)
000000C8        # third X (200)
FFFFFE0C        # third Y (-500)
F8              # A661_END_SYMBOL_BLOCK
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
# Layer Definition starts here:
A0              # A661_BEGIN_LAYER_BLOCK
05              # layer ID (5)
0000            # context number
0000008C        # block size (140 bytes)

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A1F0            # widget type (A661_PANEL)
1221            # widget ID
0000            # parent ID (zero indicates layer is parent)
01              # enable, value: A661_TRUE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001DC4        # size_x, value: 7620
0000175A        # size_y, value: 5978
0000            # style_set
0000            # UNUSED PAD
CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1222            # widget ID
1221            # parent ID
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
000002EE        # pos_x, value: 750
00000956        # pos_y, value: 2390
000017EE        # size_x, value: 6126
000009BA        # size_y, value: 2490
0000            # style set
9870            # picture reference
CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A310            # widget type (A661_SYMBOL)
1223            # widget ID
1221            # parent ID
01              # motion allowed, value: A661_TRUE
01              # visible, value: A661_TRUE
00000EE2        # pos_x, value: 3810
00000956        # pos_y, value: 2390
0000238C        # rotation angle, value: 9100
0001            # style set, value: OEM_STYLESET_FREE_COLOR
0065            # symbol reference
0F              # color index, value: OEM_WHITE
000000          # UNUSED PAD

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A310            # widget type (A661_SYMBOL)
1224            # widget ID
1221            # parent ID
01              # motion allowed, value: A661_TRUE
01              # visible, value: A661_TRUE
00000D52        # pos_x, value: 3410
00000A82        # pos_y, value: 2690
0000071C        # rotation angle, value: 1820
0001            # style set, value: OEM_STYLESET_FREE_COLOR
0064            # symbol reference
01              # color index, value: OEM_GREEN
000000          # UNUSED PAD
C0              # A661_END_LAYER_BLOCK
000000          # UNUSED PAD
E0              # A661_DF_FOOTER
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-7 XML DF Specifying Symbol Graphical Definitions

This is the XML encoding of the example given in Section C.6. Refer to Section 6 for a description of the XML DF format. **In this example not all property values are given explicitly. The unspecified properties take default values.** The WidgetId values are provided in hex format in this example. Both hex and decimal formats for unsigned integral values are allowed.

Note: the DOCTYPE specification is ad hoc. It requires that a file defining the DTD be found in the current directory with the name “a661.dtd”.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="2" supp_version="2">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <symboltable>
      <symboldefn name="HourGlassSymbol">
        <model>
           <prop name="Id" value="100" />
         </model>
         <stdrepr>
            <symboldefncmd type="A661_SYMBOL_DEFN_POLYLINE">
               <model>
                  <prop name="NumVertices" value="4" />
                  <prop name="Closed" value="A661_TRUE" />
                  <arrayprop name="Vertices">
                     <xyentry x="-500" y="-500" />
                     <xyentry x="500" y="500" />
                     <xyentry x="-500" y="500" />
                     <xyentry x="500" y="-500" />
                  </arrayprop>
               </model>
            </symboldefncmd>
         </stdrepr>
      </symboldefn>
      <symboldefn name="VertArrowSymbol">
         <model>
           <prop name="Id" value="101" />
         </model>
         <stdrepr>
            <symboldefncmd type="A661_SYMBOL_DEFN_LINE">
               <model>
                  <prop name="PosXStart" value= "0" />
                  <prop name="PosYStart" value="-1000" />
                  <prop name="PosXEnd" value="0" />
                  <prop name="PosYEnd" value="-500" />
               </model>
            </symboldefncmd>
            <symboldefncmd type="A661_SYMBOL_DEFN_TRIANGLE">
               <model>
                  <prop name="Filled" value="A661_FALSE" />
                  <prop name="PosX" value="-200" />
                  <prop name="PosY" value="-500" />
                  <prop name="PosX2" value="0" />
                  <prop name="PosY2" value="0" />
                  <prop name="PosX3" value="200" />
                  <prop name="PosY3" value="-500" />
               </model>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
            </symboldefncmd>
         </stdrepr>
      </symboldefn>
   </symboltable>
   <a661_layer name="SYMBOL_EXAMPLE_LAYER">
      <model>
         <prop name="LayerId" value="5"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="20000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="Panel" type="A661_PANEL">
         <model>
            <prop name="WidgetId" value="0x1221" />
            <prop name="PosX" value="11000" />
            <prop name="PosY" value="12700" />
            <prop name="SizeX" value="7620" />
            <prop name="SizeY" value="5978" />
         </model>
         <a661_widget name="TemperatureCelciusScale" type="A661_PICTURE">
            <model>
               <prop name="WidgetId" value="0x1222" />
               <prop name="PosX" value="750" />
               <prop name="PosY" value="2390" />
               <prop name="SizeX" value="6126" />
               <prop name="SizeY" value="2490" />
               <prop name="PictureReference"
                     value="SymbolTemperatureCelciusScale" />
            </model>
         </a661_widget>
         <a661_widget name="Symbol1" type="A661_SYMBOL">
            <model>
               <prop name="WidgetId" value="0x1223" />
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="9100" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_WHITE" />
               <prop name="SymbolReference" value="VertArrowSymbol" />
            </model>
         </a661_widget>
         <a661_widget name="Symbol2" type="A661_SYMBOL">
            <model>
               <prop name="WidgetId" value="0x1224" />
               <prop name="PosX" value="3410" />
               <prop name="PosY" value="2690" />
               <prop name="RotationAngle" value="1820" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_GREEN4" />
               <prop name="SymbolReference" value="HourGlassSymbol" />
            </model>
         </a661_widget>
      </a661_widget>
   </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-8 Binary DF Specifying Picture Definitions

This section shows a binary Definition File that defines two pictures (bitmaps). See Section C.9 for an XML version of this example.

```
# Binary Definition File
# Example demonstrates how a block of pictures can be defined.

A661            # A661_DF_MAGIC_NUMBER
01              # Library_Version
03              # Supp_Version
6788            # Application Identifier
0000            # UNUSED PAD

# Picture Definitions start here:

BB              # A661_BEGIN_PICTURE_BLOCK
C1              # PixelFormat, value: A661_PIX_FMT_RGBA_8
02              # NumberOfPicturesInBlock, value: 2
00              # NumberOfColorTableEntries, value: 0 (CDS Global Color Table)
00              # ColorTableFormat, value: 0  (ignore)
000000          # UNUSED PAD

BB              # A661_BEGIN_PICTURE
00              # UNUSED PAD
0200            # picture reference, value: 512
0100            # NumberOfPixelsWidth, value: 128
0100            # NumberOfPixelsHeight, value: 128
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
BD              # A661_END_PICTURE
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
BB              # A661_BEGIN_PICTURE
00              # UNUSED PAD
0201            # picture reference, value: 513
0100            # NumberOfPixelsWidth, value: 128
0100            # NumberOfPixelsHeight, value: 128
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
BD              # A661_END_PICTURE
000000          # UNUSED PAD

00000000        # RESERVED
BE              # A661_END_PICTURE_BLOCK
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
# Layer Definition starts here:
A0              # A661_BEGIN_LAYER_BLOCK
05              # layer ID (5)
0000            # context number
0000008C        # block size (140 bytes)

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A1F0            # widget type (A661_PANEL)
1221            # widgetID, value: 0x1221
0000            # parented, (zero indicates layer is parent)
01              # enable, value: A661_TRUE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
0000            # style_set
0000            # UNUSED PAD

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1222            # widgetID, value: 0x1222
1221            # parentID, value: 0x1221
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
0000            # style set
0200            # picture reference, value: 512

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1223            # widgetID, value: 0x1223
1221            # parentID, value: 0x1221
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
0000            # style set
0201            # picture reference, value: 513

000000          # UNUSED PAD
C0              # A661_END_LAYER_BLOCK
000000          # UNUSED PAD
E0              # A661_DF_FOOTER
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-9 XML DF Specifying Picture Definitions

This is the XML encoding of the example given in Section C.8. Refer to Section 6 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values. The WidgetId values are provided in hex format in this example. Both hex and decimal formats for unsigned integral values are allowed.

Note: the DOCTYPE specification is ad hoc. It requires that a file defining the DTD be found in the current directory with the name "a661.dtd".

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="1" supp_version="3">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <picturetable>
      <picture_defn name="ScalePicture”>
        <model>
           <prop name="PictureReference" value="512" />
           <prop name="ImageFile" value= "..\pictures\scale2.png" />
           <prop name="PictureFormat" value="A661_PIX_FMT_RGBA_8" />
         </model>
      </picture_defn>
      <picture_defn name="ScaleBackgroundPicture">
        <model>
           <prop name="PictureReference" value="513" />
           <prop name="ImageFile" value= "..\pictures\scale_background1.png"/>
           <prop name="PictureFormat" value="A661_PIX_FMT_RGBA_8" />
         </model>
      </picture_defn>
   </picturetable>
   <a661_layer name="PICTURE_EXAMPLE_LAYER">
      <model>
         <prop name="LayerId" value="5"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="25000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="ExampleScale" type="A661_PANEL">
         <model>
            <prop name="WidgetId" value="0x1221" />
            <prop name="PosX" value="11000" />
            <prop name="PosY" value="12700" />
            <prop name="SizeX" value="4096" />
            <prop name="SizeY" value="4096" />
         </model>
         <a661_widget name="Scale" type="A661_PICTURE">
            <model>
               <prop name="WidgetId" value="0x1222" />
               <prop name="PosX" value="11000" />
               <prop name="PosY" value="12700" />
               <prop name="SizeX" value="4096" />
               <prop name="SizeY" value="4096" />
               <prop name="PictureReference" value="512" />
            </model>
         </a661_widget>
         <a661_widget name="Pointer" type="A661_PICTURE">
            <model>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                    <prop name="WidgetId" value="0x1223" />
                    <prop name="PosX" value="10000" />
                    <prop name="PosY" value="12700" />
                    <prop name="SizeX" value="4096" />
                    <prop name="SizeY" value="4096" />
                    <prop name="PictureReference" value="513" />
                </model>
            </a661_widget>
        </a661_widget>
    </a661_layer>
</a661_df>
```

APPENDIX D
EXAMPLE OF 'IN/OUT" WIDGET MANAGEMENT USING STYLESET PARAMETER

Note: This appendix deleted by Supplement 3.

APPENDIX E
MAP MANAGEMENT TUTORIAL

E-1 Examples of Parameters Definition for Map Management

- ND as the master application –

  The ND has two basic modes for displaying data. The first one corresponds to Rose or Arc mode where the aircraft representation does not move on the display. The second corresponds to the Plan mode in which the aircraft representation moves and the display is centered on a point which can be very far from the aircraft. Moreover, the ND can represent the data Track-Up, (magnetic or true), Heading-Up (magnetic or true) or North-Up (typical for plan mode).

- FMS as the master application –

  The master application must first provide a projection reference point (PRP). The PRP will be used by the CDS to know what reference has to be used to run the projection algorithm. Due to the PRP, the CDS will be able to convert latitude/longitude data into a Cartesian coordinate system, for instance, true north oriented and distances in nautical miles.

  If the master application provides:

  - The position of the PRP on the display
  - The orientation of the True North relative to the Up direction of the display
  - A Range information in nautical miles and its correspondence in screen unit

  the CDS is then able to put the FMS map data on the display.

- TCAS as the master application –

  The TCAS transmits its data as bearing/distance relative to the aircraft with distance to the aircraft and bearing relative to the aircraft main axis. By adding an enumerated value in the “coordinate system” parameter of the MAPHORZ_SOURCE, that means bearing/distance relative to aircraft axis, the CDS will not have enough information to depict the traffic data for TCAS. The CDS needs the following additional data:

  - The location of the aircraft on the display
  - The orientation of the aircraft relative to the display up direction

  There is an aircraft location issue: Aircraft location is set by the master application through the MapHorz. There is an aircraft orientation issue. The master application provides the true heading through the MapHorz.

- The CDS will implement a bearing/distance MapItem relative to the aircraft. Additional parameters for the MapHorz are:
  - aircraft orientation: relative to the True North
  - aircraft latitude
  - aircraft longitude

E-2 Addressing MapItems

Inside a MapHorz_ItemList one or several MapItems can be modified through a SetParameter command with “A661_BUFFER_OF_MAPITEM” as Parameter_Ident. A MapItem will be modified in its entirety; for instance, the latitude of a symbol can not be changed by itself. But because the parameter list of each MapItem is reduced to the useful information only, all the parameters should be set in each SetParameter command.

E-2.1 Addressing MapItems for One Change:

Example A661_ParameterStructure for the SetParameter command for a SYMBOL_GENERIC:

Parameter name	Parameter value	Description
Parameter_ident	A661_BUFFER_OF_MAPITEM	Modification type
ClearFlag	0	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
Number of Items	1	Number of modified Items

ItemStructure =
{
ItemIndex
EndFlag
ItemType
SymbolType
X
Y
}
	
	{
	12
	0
	SYMBOL_GENERIC
	SYMBOL_VOR
	Longitude
	Latitude
	}
		Parameters of the modified Item.

Note that the same item number could be used once for a SYMBOL_GENERIC and later for another type of MapItem. The CDS must have enough space for the number of items specified using the biggest possible size of parameter list.

E-2.2 Addressing MapItems for Multiple Changes

Another command would be provided to access multiple Items in one command. The A661_ParameterStructure for the SetParameter command would look like the following:

Parameter name	Parameter value	Description
Parameter_ident	A661_BUFFER_OF_MAPITEM	Modification type
ClearFlag	0	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
Number of Items	2	Number of modified Items

ItemStructure =
{
ItemIndex
EndFlag
ItemType
SymbolType
X
Y
 	
ItemIndex
EndFlag
ItemType
X
Y
}
	
	{
	12
	0
	SYMBOL_GENERIC
	SYMBOL_VOR
	Longitude
	Latitude
	
	20
	0
	A661_LINE_START
	Longitude
	Latitude
	}
		Parameters of the modified Items.

Note: The set Parameter command can contain a different type of MapItem.

E-2.3 Removing Map Items

A specific ItemType is A661_NOT_USED. The parameter list for this item would be reduced to the ItemType. This approach declares a previously used Item as not used anymore without faking a type and setting its visibility to HIDE.

The A661_ParameterStructure for the Set Parameter command follows:

Parameter name	Parameter value	Description
Parameter_ident	A661_BUFFER_OF_MAPITEM	Modification type
ClearFlag	0	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
Number of Items	1	Number of modified Items

ItemStructure =
{
ItemIndex
ItemType
}
	
	{
	12
	A661_NOT_USED
	}
		Parameters of the modified Item.

E-3 Address 'Race Condition' on Item Transmission

The Map UA should handle with care the functional data associated with the dynamic widgets. Changing the functional information associated with a visible widget could cause the race condition. An example of a typical race condition follows:

- Pilot desire is to select PARIS waypoint
- At the time the pilot clicks PARIS waypoint, data is carried by the MapHorz_ItemList identified by 201 and the Item 32
- CDS sends back the event "Widget 201, Item 32" selected

In the mean time, the FMS has changed the information associated with "Widget 201, Item 32," which now carries "NEW YORK".

The problem is that the FMS cannot decide what the event truly means: has the pilot has selected PARIS or NEW YORK?

To address this problem, the FMS could have several solutions. One solution is to manage the Context Number. The UA can change the Context Number by changing the functional information attached to a widget or simply to an Item. In this way, the FMS will have the knowledge of the selected waypoint by correlation between the Context Number and the ident of the selected waypoint.

E-4 Dynamic Priority Management inside MapHorz_ItemList

The Items inside a MapHorz_ItemList are defined at run-time. The order of the Item inside the MapHorz_ItemList defines the drawing order of the items defined by the ItemIndex parameter of the item. The Item with the highest ItemIndex has the highest drawing priority. Figure E.4-1 illustrates an example of dynamic priority management.

The UA induces a specific drawing order of symbology by ordering the item inside a MapHorz_ItemList. If the UA does not specify the drawing order, then the drawing order should be defined statically in the DF with different MapHorz_ItemList for the set of symbols and drawing orders. Nevertheless, this different set of symbols corresponds to different functional sets, for which it should define a different widget (MapHorz_ItemList).

However, the drawing priority of an Item is defined by the ItemIndex of the Item. Nevertheless, the ItemIndex of an Item is independent of the transmittal order of the Item in the command. In Figure E.4-1, the UA, for instance the FMS, may have to respect a transmitting order. But the transmitting order is independent of the order of the ItemIndex declaration inside the SetParameter command.

APPENDIX E
MAP MANAGEMENT TUTORIAL

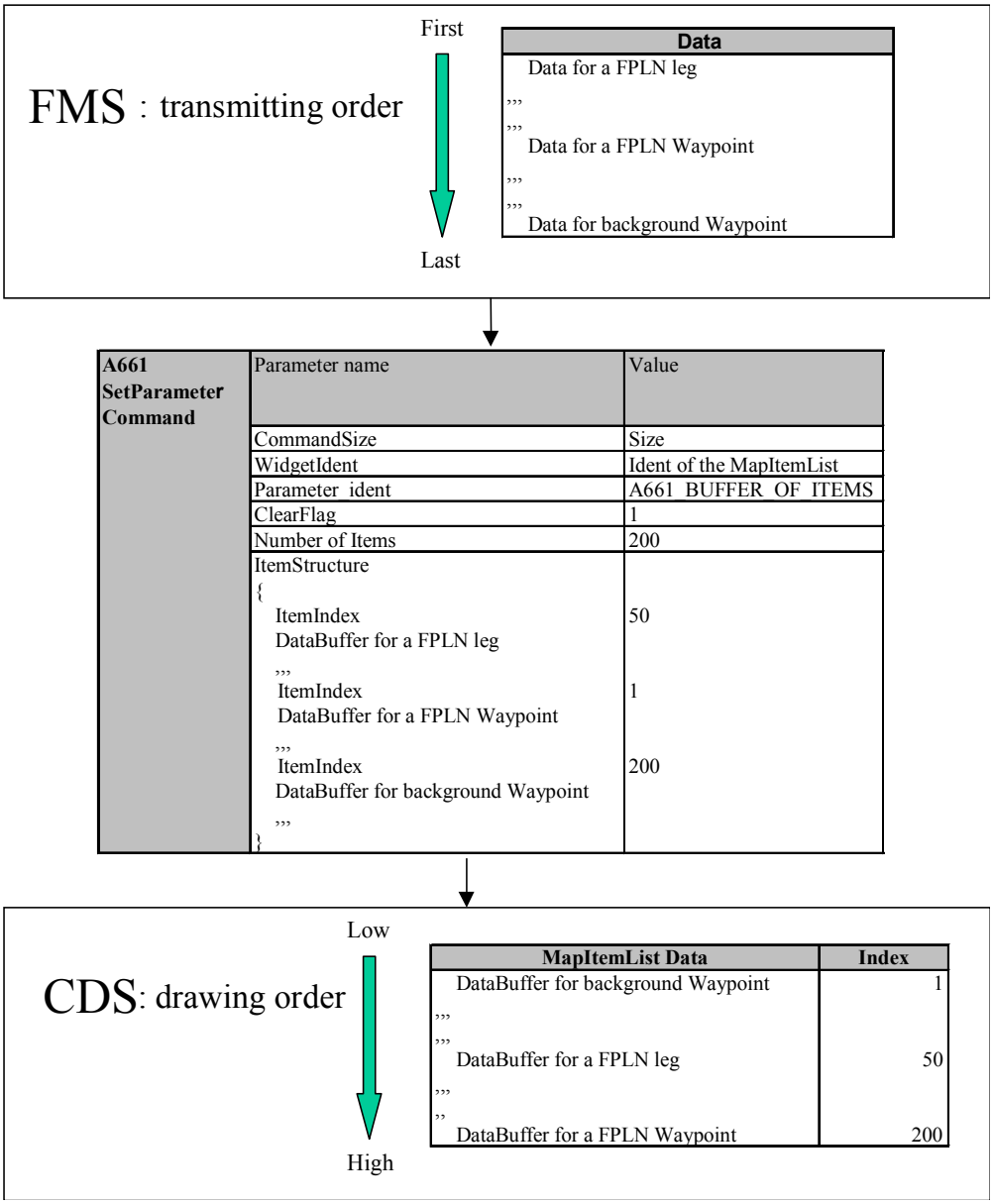


Figure E.4-1 – Example of Dynamic Priority Management

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS

F-1 Introduction

ARINC 661 defines data communication in the run-time format exchanged between User Applications (UAs) and the CDS. However, it does not specify the transport mechanism for such run-time messages. This was deliberately excluded in early versions of ARINC 661 because the computing environment and networking infrastructures used in aircraft covers such a broad range – from ARINC 429 interfaces to Ethernet derivatives, from simple real-time executives to partitioned operating systems with inter-partition communications mechanisms. Specifying exactly how ARINC 661 run-time messages are delivered to a receiver seemed to be too restrictive. Thus, this detail is left as an exercise for the implementer and the aircraft system integrator.

With this background, this appendix provides an overview of how ARINC 661 communications can work in a several typical environments. It makes suggestions on how to deal with specific problems that may arise. It is expected that the information in this appendix will be helpful to system architects that want to consider the use of ARINC 661. Following the scenarios and suggestions described herein is not mandatory to achieve compliance with ARINC 661.

F-2 ARINC 661 in Typical Aircraft Environments

Before providing specific examples and solutions, this section provides a brief summary of communication options that ARINC 661 display system designers may wish to consider.

F-2.1 Local / External Applications

This appendix describes communication transport protocols. It assumes the CDS and the UA are not located in the Line Replaceable Unit (LRU). Thus the UA is considered an external application.

ARINC 661 also allows UA and CDS to be located in the same LRU. In the case, ARINC 661 run-time messages are exchanged only within this one LRU. This approach may be chosen for design reasons (e.g., reuse of existing ARINC 661 application code, use of ARINC 661-based modeling or other development tools) or to provide for future growth that allows external UAs to use the CDS.

This case will likely have an implication for systems design: a communications mechanism between the UA and the CDS components that provides both high integrity and high reliability exists within this LRU. The following sections do not address architecture-dependent communications inside of an LRU.

F-2.2 Interactivity Impact on Communication

Interactivity in flight deck formats means that bi-directional communications between the CDS and the interactive UAs is required, such that widget events can be communicated to the CDS.

While the communications link may be bi-directional in many cases even for applications that do not display interactive formats, systems in which a UA "blindly" transmits run-time messages to the CDS are certainly possible. Other CDS-generated messages (such as error messages, notifications, etc.) can then not be sent to the UA, but that may be acceptable under the right circumstances.

F-2.3 Communication Type

Communication between CDS and UA can be periodic or event based:

- Data sent by a UA (periodic)

  Traditionally, display systems have received periodic updates (at various rates) of all aircraft data needed for the display. While data updates will likely continue to be periodic for many applications, for the communications link, having an application send periodic updates of all appropriate widget parameters means that the loss of an individual run-time message every now and then due to communications problems is acceptable.

- Data sent upon event (non periodic)

  ARINC 661 gives UAs a choice to send widget parameters only when values have actually changed. Doing this leads to software that resembles PC-based software applications more than the traditional avionics display software; however, there may be cases where that is appropriate, especially for interactive, "PC like" applications that are ported to an ARINC 661 display system.

  If UAs base the generation of some or all of their run-time messages on changed values, then the run-time messages must be sent such that they will arrive intact at the receiver with a very high probability, and a transport mechanism that can guarantee the safe delivery of information must be chosen.

  Depending on how UAs and the CDS communicate, providing that type of transport mechanism can be a non-trivial task. This may lead designers to quickly dismiss the idea of sending any widget parameters only when data changes, and instead send all necessary parameters periodically, all the time. This, however, does not completely solve the problem, because many UAs require a guaranteed delivery of widget events, notifications and other CDS-generated messages that, due to their nature, cannot just be sent to the UA many times. Once this problem is solved, the solution can be applied to messages traveling the other way, too, meaning that UAs can use the same mechanism that guarantees delivery of event messages for widget parameter updates that are not periodic.

- **Hybrid designs**

  Many hybrid designs are possible: displays that are driven by both ARINC 661 applications and the traditional ways (for different formats or different features within a format); display LRUs that contain some of the UAs but have network interfaces to communicate with others; some UAs sending ARINC 661 run-time messages blindly to a CDS while other UAs maintain bi-directional communications with the CDS, to name just a few of the possibilities. ARINC 661 does not prevent (or even discourage) to mix and combine different solutions where that makes sense.

The following sections show how ARINC 661 can work in selected, typical environments. Detailed discussions of industry terms (such as Ethernet) and other ARINC specifications are not provided here; there is no lack of other information about these terms and concepts for the interested reader.

F-3 Ensuring Reliability

For the purposes of this discussion, the term "reliability" means the probability that a message sent by some transmitting application is received by the receiving application. Typical network connections (such as Ethernet, ARINC 429, etc.) drop data every now and then, so they are not reliable unless some mechanism is added at a higher level to deal with dropped data. This is not to be confused with the network's integrity, which describes the probability that a message that arrives at the receiving application contains has not been accidentally corrupted (without detecting the) during the transmission. In other words, high integrity ensures correctness of received messages, while high reliability makes sure takes care of completely receiving all sent information.

Data connections used in avionics systems typically have good integrity but relatively poor reliability. There are three different ways to deal with the reliability issues:

- **Periodic Transmissions:**

  Parameters are sent repeatedly at some periodic rate. Examples where this is typically done include aircraft attitude, airspeed and altitude data. The loss of a single piece of data is not significant because the transmitter will send new values anyway. The probability that two, three, or more subsequent instances of a data value are all lost is much smaller than the probability to lose a single value, and for this reason, low network reliability is typically not an issue if periodic transmissions are used.

- **Acknowledgements (ACK/NACK):**

  Acknowledgement-based protocols require that the receiver sends a confirmation to the sender for information that it has received. That way, the loss of a data packet is recognized and the sender can send another copy of the original data

over the network. These protocols exist in various different forms. Some widely known examples are TCP and TFTP.

- Sequence Numbers:

  Another mechanism to detect the loss of data packets is to have the receiver add a sequence number to each packet, which the receiver monitors. The receiver expects that subsequent messages have adjacent sequence numbers; if it receives anything else, it can assume that a message has been dropped, and some corrective action can be initiated.

Note that these concepts are not mutually exclusive.

F-4 Ethernet

Ethernet and UDP-based network connections (such as AFDX) are generally a good choice for ARINC 661, because they offer a high bandwidth along with good integrity. They are also bi-directional. However, Ethernet is not, by itself, reliable, so if a guaranteed delivery of run-time messages is required, an additional protocol level above UDP or IP is required.

Several industry standards exist that make Ethernet more reliable. The two most widely used ones are TCP and TFTP. Both TCP and TFTP require the receiver to send a confirmation for received messages. If a confirmation is not received back at the sender within a certain amount of time, the sender will retransmit the information.

Both TCP and TFTP can be useful to ensure reliability for ARINC 661 run-time messages. A few things should be considered carefully, though:

- With TCP, sender and receiver determine the size of a “window”, and the sender is allowed to send as much information as will fit in this window until a receipt for a portion or all of the window’s data is received. Thus, TCP does not generally block the sending application – it only does so if the entire window lacks an acknowledgement through the receiver.

  Advantages:

  - Transmit run-time message reliably over an unreliable network

  Drawbacks:

  - TCP is a very complex protocol. It was created to solve transport problems in the world-wide Internet, and many of those problems do not really exist on aircraft networks. If provided on an aircraft network, TCP can be used for ARINC 661, but its complexity and overhead should be kept in mind.
- In TFTP, each message sent by a transmitting application must be confirmed by the sender before more information can be sent. In case of a communication problem, TFTP will

prevent a sender from sending new messages until the problem has been recovered by a re-transmission. In case of re-transmission failure, a high level communication sanction has to be taken, such as Communication re-initialization.

Advantages:

- Transmits run-time message reliably over an unreliable network
- It allows latency and network bandwidth optimization (by sending only once the message).
- This communication protocol is well-known (and thus the associated drawback effect).

Drawbacks:

- Blocking nature of the TFTP, UA application has to receive the message acknowledge before sending new data. Thus a particular attention shall be paid to TFTP and network configuration. Besides, re-transmission could lead to additional latency in degraded cases.
- For connections that carry periodic data, this is probably a poor choice, whereas for certain interactive, “PC like” formats it can work well.

F-5 ARINC 429

ARINC 429 is a serial interface that is widely used for exchanging information on aircraft, including for display systems. It is unidirectional and does not provide a guarantee for delivery of sent messages. Its wide availability on aircraft makes it an interesting consideration for ARINC 661, though.

ARINC 661 run-time messages can be sent over ARINC 429 connections. The result is somewhat similar to Ethernet/UDP, operating at much slower data rates, though. If reliable communications is needed, an intermediate protocol between ARINC 429 and the ARINC 661 run-time messages can be used, similar to TFTP and TCP (as discussed above).

F-6 Example Protocol for Reliable Communications

This reliable communications protocol is based on sequence number. It is suggested as an optional mechanism for ARINC 661 display systems, and can serve as a starting point when circumstances require the use of such a mechanism.

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS

The main goals for the design of the reliable communications mechanism are:

- **Transmit run-time message reliably over an unreliable network**
- **Determinism – achieve predictable network load and behavior**
- **Include status (health, availability) information into the design**
- **Compatible with multicast/broadcast transmission**

The protocol works mostly by sending out multiple copies (spread out over time) of data that a sender considers important, whereas other data is sent only once, such as data that is generated cyclically anyway and for which a lost value every now and then would not cause a problem.

Drawback effect of this proposal could be:

- **Bandwidth use of the network: Need to transmit several times a set of data depending of network reliability.**
- **Next message is blocked until all the multiple copies of the previous message have been sent.**

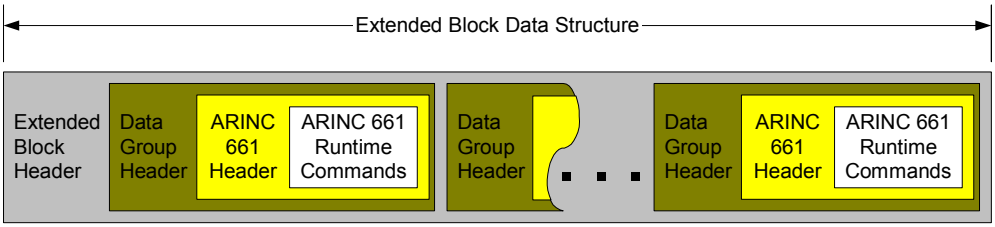
The mechanism consists of two levels of wrappers around ARINC 661 run-time data: a "group" and a "block". Conceptually, a "block" can be thought of something that an application transmits (at least) one of per cycle. It is atomically transmitted by the network in that it is either received and undamaged, or not visible to the receiver at all. A block consists of the block header and zero or more "groups".

A "group" consists of one or more ARINC 661 run-time message blocks. Anytime a UA or a CDS intends to send an ARINC 661 run-time message block, that block is wrapped in a "group". "Groups" can be sent reliably or unreliably, which is a choice the sender makes.

The sender assigns a sequence number to each group. Groups that carry unreliable data have a sequence number of zero. Groups that carry reliable data have a sequence number greater than zero. The sender starts with a sequence number of one for the first reliable group, and increments it by one each time a new reliable group is generated. Copies of a reliable group have the same sequence number as the first version of that group.

The Groups within an Extended Block are assembled so that they are in order of sequence number, from lowest to highest, except that groups with a sequence number of zero (i.e., unreliable groups) may appear anywhere within an Extended Block.

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS



Extended Block

The Extended Block is structured as defined in Table F-6.1.

Note: The first 12 bytes are the Extended Block Header.

Table F-6.1 – Extended Block Structure

Offset (bytes)	Size (bits)	Parameter	Description	Value/ Type
0	16	Start Marker	Indicates the start of the Block Header.	"BS" 0x4253
2	16	Extended Block Size	Size in bytes of the extended block including the header.	0-65535
4	4	Source	Identifies the CDS or UA originator of the message block.	0-15
	4	Destination	Identifies the CDS or UA to which the message block is destined.	0-15
5	8	Number of Groups	Number of groups in this extended block.	0-255
6	8	Service Available	Used by UAs to indicate whether the service associated with this connection is available for use by the CDS.	True/ False
7	8	Assumed Health	For each connection, the CDS and UAs compute health of their counterpart and echo this assumed health in subsequent transmissions.	True/ False
8	32	Lowest Sequence No.	Used by the reliable communications protocol to indicate the lowest sequence number of data that is being or will be sent.	
12	{32}+	Extended Block Data	The extended block data is made up of any number of Data Groups.	

The values for "Source" and "Destination" are meant for multi-cast scenarios, where more than one UA or CDS listens on any given connection. In this case, a number between 0 and 15 is assigned to each sender and each receiver on each connection. These fields can be ignored if point-to-point connections are used between each CDS and each UA.

Both UAs and the CDS exchange extended blocks periodically to exchange health information. If a UA or a CDS determines that its own health is bad, it should stop sending periodic extended blocks. Extended blocks may contain any number of groups, including none if there is no actual data to send and the purpose of the extended block is merely to announce the sender's health.

The intended use for the "Service Available" parameter means that it should be set either by a CDS or a UA. The idea is that either a CDS picks a UA to drive a layer, or a UA picks a CDS to display its layers. In either case, the "Service Available" flag is set to True if the sender is able to support a new request from the receiver.

The "Assumed Health" parameter is used to tell the receiver what health it assumes for that receiver. For example, if a CDS is communicating to a UA, the CDS will determine the health status of that UA (either Failed or Valid). When the CDS transmits data to UA it will set the value of "Assumed Health" in the Extended Block Header to the value of the Health Status it computed for the "UA". This helps address problems that might otherwise arise from a failure in just one way of a bi-directional connection.

The use of the Lowest Sequence Number parameter is defined below (see "Reliable Communications Method").

Data Groups

The Data Group shall be structured as defined in Table F-6.2 below.

Note: The first 8 bytes are the Group Header.

Table F-6.2 – Data Group Structure

Offset	Size	Parameter	Description	Value
0	16	Start Marker	Indicates the start of the Data Group Header.	"GS" 0x4753
2	16	Data Group Size	Size of the group in bytes (incl. the group header).	
4	32	Sequence Number	Sequence Number for the group.	
8	{32}+	Group Data Block	Contains one or more ARINC 661 data blocks and a maximum of one control block.	

The Sequence Number is defined in the following section.

Reliable Communications Method

The reliable communication method allows a CDS or UA to transmit a message block multiple times and allow the receiving system a means of determining whether it has missed one of these blocks.

Note: The number of times the transmitter sends the block is variable and can be configured to achieve the desired reliability across the network. The assertion being that sending the

block multiple times increases the probability that one of the blocks will arrive.

Transmitting System

When a connection is first established, the transmitting system sets the value of Transmitted Sequence Number to zero. Each time the sender creates a fresh block of new data that needs to be transmitted it increments the Transmitted Sequence Number by one and sets the Sequence Number in the Group Header to this value.

Note: If a group is to be sent multiple times, then each duplicate message will have the same sequence number.

If the sender wishes to send a non-reliable block of data it sets the Sequence Number in the Group Header to zero.

The sender sets the value of Lowest Transmitted Sequence Number to the lowest non-zero sequence number for a message block that is still scheduled for re-sending, or the next future sequence number if there is nothing more to send.

The sender transmits this group in successive message blocks until it has been sent the number of times required to assure reliable transmission.

Receiver

When a connection is first established, the receiver sets the value of Expected Next Sequence Number to one.

The receiver compares the value of Expected Next Sequence Number to the Sequence Number value in each Group Header for new message blocks. If the Sequence Number in the Group Header is equal to the Expected Next Sequence Number then the receiver processes the associated block of ARINC 661 data, increments the Expected Next Sequence Number by one and then process the next group.

If the Sequence Number in the Group Header is not equal to the Expected Next Sequence Number, then the receiver skips to the next group.

If the Lowest Sequence Number in the Extended Header is higher than the Expected Next Sequence, the receiver knows that it has experienced an unrecoverable loss of data. To resynchronize, the receiver then sets the Expected Next Sequence Number to the Lowest Sequence Number in the Extended Header and then asserts the “Data Lost” state. If the receiver is a CDS, it sends the A661_NOTE_REINIT_LAYER to the associated application. If the receiver is a UA, it reinitializes all layer data as if it had received an A661_NOTE_REINIT_LAYER message. And the data is recovered.

APPENDIX G
NEW WIDGET GUIDELINES

G-1 Guideline Purpose

The purpose of the New Widget Guidelines is to:

- Assist candidate widget advocates in successfully proposing new widgets.
- Assist reviewers in a conducting a structured evaluation of proposed new widgets.

The proposal of new widgets can be divided into two general parts.

1. The Description / Discussion section that provides a higher level discussion of the proposed widget's purpose.
2. The Template section that provides the detailed widget attributes that will ultimately be included in the ARINC 661 specification.

G-2 Description/Discussion

Potential new widgets should be introduced with a high level discussion that addresses why the proposed widget should be added to the Widget Library. As applicable, that discussion should include the following points:

- Describe the new widget's purpose and operational capability it provides for the aircraft. If possible, provide a representative figure and illustrative example. If applicable, describe how the user interface functions. Where does the new widget fit in the A661 library structure?
- If applicable, explain why the current library is insufficient to accomplish the requirement. What does the candidate widget do that can not be reasonably accomplished with the current widget library?
- Is this new widget an extension of a current capability or a fundamentally new capability?
- How does this widget interface with other widgets?
- Could a current widget be modified to accomplish the intended purpose? If so, why is a new widget a better idea? How does this proposed widget differ from related widgets (what attributes are different). Why is the new widget better than a previous widget?
- Are there any backward compatibility issues with this widget or widget change?
- Does this widget make any current widgets obsolete and candidate for removal from the standard?
- Does the new widget design include any growth capability?
- Are there likely to be any processing or bandwidth resource issues with the new widget?

- **Is the definition of widget parameters independent of "look and feel" concerns that could be better addressed by StyleSet.**
- **Is the new widget best suited as a standard A661 library object or is it more appropriately implemented as an OEM custom widget?**
- **If the introduction of the new widget introduces new concepts beyond those already defined in ARINC 661, the advocate should include material that describes the advantage of the proposal and how it might be specified in ARINC 661.**

G-3 Widget Design Guidelines

The following guidelines should be considered when designing a new widget. If a widget does not comply with these guidelines, an explanation should be provided.

Each rule is associated with an example to illustrate the way it should be understood (example of good usage of the rule or of violation of the rule).

Note: For examples of rules violation, the provided examples have been discussed.

1. **Each widget should have a single role / responsibility. If a widget has several roles (for instance depending on a parameter), it should be split into a set of simpler widgets having a single role.**
   A widget could be a scrolled-list or a pop-up list depending on its size (still allowing to select one item out of several), but a widget can not be a pop-up list (selecting an item out of several) and a text-field (entering a text) depending on the value of its parameters.
2. **Each parameter should have a single goal. If a parameter has several goals (for instance depending on other parameters), it should be split into a set of simpler parameters having a single goal.**
   Example of rule violation: ExternalSource SourceDX and SourceDY parameter have different behavior depending on the Source parameter.
3. **A widget should not rely on hypothesis on its container.**
   Example of rule violation: the tabbedPanel widget can only be the children of a tabbedPanelGroup widget.
4. **Widget parameters should be individually modifiable: provided a parameter value is set consistently with its usage domain, setting this value should not result into an erroneous behavior due to the value of other parameters.**
   Example of rule violation: the AlphaMask and NumericMask value can not be changed if not consistent with the labelString.

5. Each layer has an associated namespace (widget ID). The parameter of a widget belonging to a layer should not use reference to the namespace of another layer (in this case, use a CDS-level reference).
   Example of rule violation: FocusLink is making a reference to a widget belonging to another layer.
6. StyleSet should not be used to describe anything but alternative views (no different controller depending on the StyleSet)
   Example of rule violation: the StyleSet should not be used to change a radiobox into a multiple selection container.
7. No widget should rely on assumption of cockpit configuration (number of users, number of cursors, size and number of displays, type of control, etc.)
8. All parameters should be definition and run-time modifiable except if they raise a safety or system issue. This analysis should be done before adding a new widget to ARINC 661.
   Example: Maximum string length is not run-time modifiable because changing the value of this parameter at run-time may result in undesired effects (memory allocation issue).
9. If a widget is derived from an existing widget, it should inherit (in term of parameters) from the existing widgets (having the same parameters plus specific parameters).
   Example: PicturePushButton is derived from the PushButton.
10. All widget should follow convention of parameter types / sizes / run-time identifiers / events defined for existing widgets. For example, all parameters related to screen dimension should be 32 bits and related to 1/100th millimeters.

G-4 New Widget Template

Section 3.3 describes ARINC 661 widget characteristics and the interface.

Each widget definition is divided into the following parts that need to be included in a new widget proposal.

1. Definition Section
   a. Categories
   b. Description
   c. Restriction
2. Widget Parameters Table
3. Creation Structure Table
4. Event Structure Table
5. Run-time Modifiable Parameter Tables

While it may not be necessary to complete the entire widget definition when initially proposing a widget it is helpful to highlight the areas that make the widget unique.

Definition Section

G-5 Widget Category

Widgets are grouped into one or more categories based on the widget's purpose. Table 3.2.2-6 is the current list of Widget Library Categories. The following is provided for convenience.

- **Container**
- **Graphical Representation**
- **Text String**
- **Interactive**
- **Map Management**
- **Dynamic Motion**
- **Utility**
- **UA Validation**

Description
Provide a functional description of the widget.

List any widgets it is similar to or dependent upon.

Restrictions
Describe any restrictions to ARINC 661 principles.

Describe any exceptions to defined provisions. For example, Sections 2.3.4.1 through 2.3.4.3 provide defined conventions for positioning and sizing within a window. Exceptions to these conventions should be identified and explained.

Widget Parameters
The parameters associated with each widget can be divided into commonly used parameters and specific parameters. Specific parameters can be either previously used or new ones. Requirements for new specific parameters will need to be clearly explained as part of the new widget proposal.

The Commonly Used Parameters List below is provided for convenience and lists common parameters and identifies the parameters that are required for all widgets. The list is an abbreviated compilation of the information found in section 3.1.3

Section 3.1.3 provides a full discussion of Commonly Used Parameters.

APPENDIX G
NEW WIDGET GUIDELINES

Pay special attention to the StyleSet parameter. It is desirable for the ARINC 661 interface to be as independent as possible from the widget's graphical characteristics. The parameter allows the different OEMs to ensure the widget graphics meets their requirements.

The following table is provided to assist in listing the new widget's parameters.

Parameters	Change	Description
Commonly Used Parameters		
		See Commonly Used Parameter list below or Section 3.1.3
Previously Used Specific Parameters		
		See Table 4.6-8 – Parameter Types
New Specific Parameters		

G-6 Commonly Used Parameters List

Note: (Req) indicates a required widget parameter

Widget Parameter	Category Description
WidgetType (Req)	Type of widget See Table 4.6.7
WidgetIdent (Req)	Identifier of the widget (refer to Section 3.1.1,) WidgetIdent is a non-null positive value. NULL is reserved for referring to the layer level (e.g., ParentIdent)
ParentIdent (Req)	Identifier of the immediate container of the widget. Only a special category of widgets called "Container" can be the parent of other widgets. At the highest level of the widget hierarchy within a layer, the ParentIdent value is 0 (NULL). This means that the parent of the widget is the layer.
InnerState	Holds the specific functional state (if any) of a widget. The set of possible values depends on widget type.
Visible	A661_FALSE: The widget will not be rendered. A661_TRUE: If its entire parent is visible, the widget will be rendered. If one or all its parents are invisible, the widget will not be rendered, whatever the value of its visible parameter.

APPENDIX G
NEW WIDGET GUIDELINES

Widget Parameter	Category Description
Enable	A661_FALSE: The widget will not be interactive. A661_TRUE or A661_TRUE_WITH_VALIDATION: If all its parents are enabled, the widget will be interactive. If one or all its parents are disabled, the widget will not be interactive, whatever the value of its Enable parameter. An invisible widget is not interactive, independent of the value of its Enable parameter.
Anonymous	A661_FALSE: run-time accessible. Widget can be modified at run-time, if it has some run-time accessible parameters. A661_TRUE: anonymous. Widget can not be modified at run-time by UA. CDS behavior when a UA attempts to SetParameter on an anonymous widget is undefined.
StyleSet	StyleSet allows the UA to select from a predefined set of graphical characteristics to be applied to a widget. See Table 3.1.3.3 for a full discussion of the StyleSet parameter.
Pos X	The X position of the widget reference point is an offset with respect to the absolute X position of the reference point of the widget container (parent).
Pos Y	The Y position of the widget reference point is an offset with respect to the absolute Y position of the reference point of the widget container (parent).
Size X	The X dimension size (width) of the widget.
Size Y	The Y dimension size (width) of the widget.
NextFocusedWidget	Widget ident of next widget to be focused upon crew member validation.
AutomaticFocusMotion	A661_FALSE: No automatic motion: after a crew member validation, the focus remains on the widget until an explicit move of the focus. A661_TRUE: Move automatically the focus after a crew member validation to the next widget according to the NextFocusedWidget parameter

G-7 Creation Structure

A Creation Structure Table should be prepared for each candidate widget.

In the Creation Structure Tables, parameters are grouped together to form 32-bit words. Each word in the table is separated from other words by a full line. When one word of 32 bits is composed of several parameters, the parts are separated in the table by a dashed line. Refer to Table 3.3-2, Example of Creation Structure.

The widget parameter order discussed in the previous section may be different from the order in the widget parameter table. The widget parameter table describes parameter functional aspect, while the creation structure table describes the parameter buffer coding aspect.

Some points to remember when generating a new Creation Structure:

- If possible, allocate a set of pad bits (16 to 32 bits) at the end of the widget creation structure for future growth.
- Ensure the StyleSet size is 16 bits.
- The Enable flag goes just before Visible flag (exceptions are the Connector and CursorOver widget).
- Use existing parameter types as much as possible, refer to tables in Section 4.
- When adding codes to the tables in Section 4, be careful to avoid adding a code to a table that has already been assigned in that table (exception: constants can be duplicated). Some checking beforehand is required before assigning a code: not all tables are ordered according to increasing code value, some are ordered alphabetically or per some other method.
- Use {n}+ notation to indicate variable sized buffer of parameters.
- Pads are all type N/A.
- Within a word, place 16-bit parameters before 8-bit parameters.

Add pads at the end of any unfilled 32-bit words.

The following table is provided to assist in completing the new widget's Creation Structure table.

Create Parameter Buffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	See Table 4.6-9, Widget Types
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
			Continue the table, ensuring all widget parameters are included.

G-8 Event Structure

Interactive widgets must have an event structure table attached. This structure is similar to the widget creation structure and provides the structure for widget associated events.

An Event Structure Table should be prepared for each Event.

In the Event Structure Tables, parameters are grouped together to form 32-bit words. Each word in the table is separated from other words by a full line. When one word of 32 bits is composed of several parameters, the parts are separated in the table by a dashed line. Refer to Table 3.3-2, Example of Event Structure.

A discussion of widget events can be found in Section 3.1.4, Widget Events.

Table 4.5.4.2-3 defines the run-time Widget_Event_Structure.

The following table is provided to assist in completing the new widget's Event Structure Table.

Event Structure	Type	Size (bits)	Value / Description
EventIdent	ushort	16	See Table 4.6-9, Event Types
As required.			See Table 4.6-10, Boolean Constants
			See Table 4.6-11, Integer Constants
UnusedPad	N/A	variable	If required

G-9 Runtime Modifiable Parameters

The dynamic data transfer between the UA and CDS at run-time includes the requirement to update run-time widget parameters. Runtime Modifiable Parameters should include all widget parameters that have "R" in the Widget Parameter Table Change column.

APPENDIX G
NEW WIDGET GUIDELINES

Section 4.5.4.5, ARINC 661 Parameter Structure, provides details of the parameter structures which should be applied to run-time modifiable parameters.

Some points to remember when generating a new runtime modifiable parameter:

- Never modify WidgetType or WidgetIdent
- Do not use StyleSet to manage race condition
- "XY" and "DXDY" are for convenience only. This allows the UA to set two adjacent parameters at once, as no corresponding parameters exist in the Creation Structure.
- Follow existing widget as much as possible when naming runtime modifiable parameters, (e.g., abbreviate longitude as 'Lng' not 'Lon')

The following table is provided to assist in completing the new widget's Runtime Modifiable Parameter table.

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)

G-10 Widget Library Tables

As part of the inclusion of a new widget in the Section 3.0, Widget Library, the tables identified below will need to be updated.

- Widget Event Use Cross Reference Table - 3.1.4-1
  This table should be updated.
- Widget Library Summary – 3.2.1
  Brief widget description.
- Widget Classification Table – 3.2.2-2
  List widget categories (Container, Map Management, Dynamic Motion, Graphical Representation, Text String, Interactive, Unclassified)
- Widget Children – 3.2.3.1
  List all possible parents of the new widget. For new container widget, list the possible children.
- ARINC 661 Keyword Values - 4.6
  Provide input to all applicable tables.

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7465

SUPPLEMENT 1

TO

ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACE TO USER SYSTEMS

Published: June 26, 2003

Prepared by the Airlines Electronic Engineering Committee

Adopted by the Airlines Electronic Engineering Committee: March 6, 2003

A. PURPOSE OF THIS DOCUMENT

This supplement introduces various changes and additions to ARINC Specification 661. It adds eight new widgets to the standard and provides clarification of the definition of the Cockpit Display System (CDS) interface to user systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document, vertical change bars in the margin will indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

0.0 Global Changes to Nomenclature Used in ARINC 661

- MapWidget changed to MapHorz
- MapSource changed to MapHorz_Source
- MapItemList changed to MapHorz_ItemList

2.2.1 Definition Phase

This section modified for clarity.

2.2.3 Special Conditions

This section and subordinate sections added.

2.2.4 ARINC 661 Conformance

This section re-numbered.

2.2.5 ARINC 661 Library Evolution

This section re-numbered.

3.2.2 Widget Classification

Table 3.2.2-2 updated to support the expansion of widgets.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 updated to support the expansion of widgets.

3.2.5.3 Change Style Capabilities

Table 3.2.5.3 corrected error by replacing “Flashing” with “Animation”. Replace the definition with “Animation of ASCII text”.

3.2.5.5 Escape Sequences Description

Table 3.2.5.5.1 and 3.2.5.5.2 corrected error by replacing “Flashing” with “Animation”.

3.2.8 Map Management

This section modified for clarity.

3.3.1 Active Area

Added StyleSet Parameter.

3.3.2 BasicContainer

This section modified to state that some widgets can only be positioned at run-time. supplement 1 supports the definition of the position of an optional panel at run-time. The objective is to define a position, not to move the widget at run-time.

3.3.5 CheckButton

This section modified to define the label alignment on button. The definition of the LabelPosition parameter is clarified. This parameter can be replaced by “PicturePosition” to be coherent with PictureXxxxButton.

3.3.6 ComboBox

This section modified to define the label alignment on button.

3.3.9 EditBoxMasked

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of ReportAllChanges changed from Boolean to Enumeration. STATE_CHANGE event deleted. ABORTED event added.

3.3.10 EditBoxNumeric

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of ReportAllChanges changed from Boolean to enumeration. STATE_CHANGE event deleted. ABORTED event added. Parameters NumericKeyFlag, MinValue, MaxValue and CyclicFlag are added. TicsCoarse and TicsFine are not modifiable at run-time.

3.3.11 EditBoxText

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of

ReportAllChanges changed from Boolean to enumeration. STATE_CHANGE event deleted. ABORTED event added.

3.3.20 Label

Changed “static” to “anonymous”. Deleted references to “blink” capability. Added “ColorIndex” parameter. Added additional Alignment value definitions.

3.3.21 LabelComplex

Added additional Alignment value definitions.

3.3.22 MapHorz_Item List

MapItemList changed to MapHorz_Item List.

3.3.22.1 MapHorz_ItemList Standard Items Description

Added “FilledPolyStart” and “FilledOval” map items.

3.3.22.2.1.8 Symbol Generic

SymbolType values were labeled as examples.

3.3.22.2.1.10 Symbol Rotated

SymbolType values were labeled as examples.

3.3.22.2.1.11 Symbol Runway

The words “coordinate of symbol” changed to “coordinate of threshold”.

3.3.22.2.1.12 FilledPolyStart

This section added.

3.3.22.2.1.13 SymbolOval

This section added.

3.3.23 MapLegacy

Parameter “FormatType” changed to “ChannelID”. All references to ARINC 702 and ARINC 708 were removed. This makes MapLegacy consistent in description and operation to ExternalSource widget.

3.3.24 MapHorz_Source

MapSource changed to MapHorz_Source. Table of MapDataFormat valued added for clarity. “A661_EVT_SELECTION” changed to “A661_EVT_SELECTION_MAP”.

3.3.25 MapHorz

This section modified to support map display. Map Widget changed to MapHorz. Description of “PRP Lat/Lng” identified as Commentary. Description of “Orientation” updated for clarity. Screen Reference Point X/Y changed from ulong to long.

3.3.28 PicturePushButton

Alignment parameter added.

3.3.29 PictureToggleButton

Alignment parameter added.

3.3.30 PopUpPanel

AutomaticClosure parameter added.

3.3.31 PopUpMenu

Replace “UAPositionFlag” by “OpeningMode”, because an UA may want to open a menu UP or DOWN.

3.3.32 PopUpMenuButton

It is necessary to define the label alignment on button. Replace “UAPositionFlag” by “OpeningMode”, because an UA may want to open a menu UP or DOWN.

3.3.33 PushButton

It is necessary to define the label alignment on button.

3.3.34 RadioBox

Updated to say that a user application may need to display a RadioBox without any selected element (for example, in the disable state).

3.3.36 ScrollPanel

HorizontalScroll and VerticalScroll modified to cover all possibilities, Absent/Up/Bottom/Left/right. For operational reasons, it might be necessary to place vertical and horizontal scroll at the same place. It is easier for a crew member to manage the scroll buttons.

3.3.37 ScrollList

It is necessary to define the label alignment on button.

3.3.38 Symbol

Category does not include “interactive”. This category was deleted.

3.3.39 TabbedPanel

It is necessary to define the label alignment on button. The UA managing a TabbedPanel (or a set of TabbedPanel) may need to define inset size. To introduce this functionality and keep the segregation between the TabbedPanel and the TabbedPanelGroup, new parameters were added. For Tabbed Panel, the “InsetSize” parameter is added. For TabbedPanelGroup, the “AutomaticInsetSizeFlag” parameter is added. This flag allows the choice between the manual inset size using “InsetSize” parameter or an inset size defined by a display dependent algorithm.

3.3.40 TabbedPanelGroup

The UA managing a TabbedPanel, or set of TabbedPanel, may need to define inset size. To introduce this functionality and to keep the segregation between TabbedPanel and the TabbedPanelGroup, new parameters were added. For Tabbed Panel, the “InsetSize” parameter is added. For TabbedPanelGroup, the “AutomaticInsetSizeFlag” parameter is added. This flag selects between the manual

inset size using “InsetSize” parameter, or an inset size defined by a display dependent algorithm.

3.3.41 ToggleButton

It is necessary to define the label alignment on button.

3.4 Widget Library Expansion

This section and its subordinate sections added by supplement 1.

3.4.1 MapGrid

This section added. New Widget is defined.

3.4.2 ExternalSource

This section added. New Widget is defined.

3.4.3 MapVert

This section added. New Widget is defined.

3.4.4 MapVert_Source

This section added. New Widget is defined.

3.4.5 MapVert_ItemList

This section added. New Widget is defined.

3.4.6 EditBoXMultiLine

This section added. New widget is defined.

3.4.7 ComboBoxEdit

This section added. New widget is defined.

3.4.8 MenuBar

This section added. New widget is defined.

4.0 COMMUNICATION PROTOCOL

This section modified to reflect changes elsewhere in the document.

4.6 ARINC 661 Keyword Values

This section updated.

APPENDIX C – EXAMPLE OF A DEFINITION FILE

The example was updated following an actual implementation in 2003.

APPENDIX E - MAP MANAGEMENT TUTORIAL

This appendix modified to provide example of a map display.

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7465

SUPPLEMENT 2

TO

ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

Published: June 30, 2005

Prepared by the Airlines Electronic Engineering Committee

Adopted by the Airlines Electronic Engineering Committee: October 27, 2004

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to ARINC 661, Cockpit Display System Interface to User Systems. The supplement introduces seven new widgets that expand the capability of ARINC 661 display systems.

To make communication consistent across all widgets, changes to the communications protocol are included for some widgets, Standardized data structure and communication protocols are introduced, with the expectation that subsequent supplements will be compatible with this version. As a result of these changes, it is recognized that the communication protocol for some widgets included herein may not be compatible with communication protocols defined in earlier versions of this standard. The intent is for all CDS implementations built to this standard and future versions of this standard to use the same communication protocols.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document, vertical change bars in the margin will indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

Global Changes to Widgets and Data Structures Used in ARINC 661

The pad bits used within the data structures were modified to follow existing conventions more consistently.

The FocusIndex attribute was replaced by NextFocusWidget

ParameterStructure_XY was replaced by ParameterStructure_8Bytes

2.3.5 Cursor Management

This section was updated to provide guidance on cursor focus and highlight. The changes are intended to better describe cursor control timing.

3.1.3.5 Parameters Related to Focus Navigation

Table 3.1.3.5 was modified to identity of the widget to be focused upon was clarified.

3.1.4 Widget Events

This Section was added to describe behavior of widget events.

3.2.1 Widgets Summary

The widget library summary was expanded to include seven new widgets introduced in supplement 2. These include:

1. MutuallyExclusiveContainer widget
2. ProxyButton widget

3. WatchdogContainer widget
4. Slider widget
5. PictureAnimated widget
6. SymbolAnimated widget
7. SelectionListButton widget

3.2.2 Widgets Classification

Table 3.2.2-2 was updated to support the widget expansion in supplement 2.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 was expanded to include new widgets and the relationship with respect to each other.

3.3 Widget List

Table 3.3-1 was updated to correct the LSB value for 32 bit words. Conventions for padding and alignment were added.

3.3.1 ActiveArea

A parameter was modified to include the identity of the widget to focus upon.

3.3.2 BasicContainer

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.4 BufferFormat

The restrictions on BufferOf Parameter were modified.

3.3.4.1 A661_ParameterStructure_Buffer

This section was expanded to provide additional detail.

3.3.5 CheckButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.6 ComboBox

This section was modified to add new OpeningEntry parameter. A parameter was modified to include the identity of the widget to focus upon.

3.3.9 EditBoxMasked

An EntryValidation parameter was added. This would clarify use of alpha and numeric characters.

3.3.10 EditBoxNumeric

The FormatString parameter was changed to be run-time modifiable. New MaxFormatStringLength parameter was added. Events reported by this widget were updated.

3.3.11 EditBoxText

FormatString parameter was changed to be run-time modifiable. New MaxFormatStringLength parameter was added. Events reported by this widget were updated. New parameters StartCursorPos and LegendString were added.

3.3.13 GpArcCircle

Runtime parameters PosX and PosY were defined to be 8 bytes. UnusedPad were changed from 8 bits to 16 bits for proper alignment.

3.3.14 GpCrown

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.15 GpLine

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.16 GpLinePolar

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.17 GpRectangle

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.18 GpTriangle

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.22.1 MapHorz_ItemList Standard Items Description

New MapItem types ITEM_SYNCHRONIZATION was added.

3.3.22.2 MapHorz_ItemList A661_ParameterStructure Specifics

The description of symbol placement was added. Also, throughout this section, RelativePosition attribute was added.

3.3.22.2.1.6 Line_Arc

This section was updated to clarify the description of InboundCourse and CourseChange angles.

3.3.22.2.1.14 Item_Synchronization

A new MapItem type was added.

3.3.22.3 MapHorz_ItemList Interactive Items

This section was added to define interactivity on maps.

3.3.24 MapHorz_Source

The EventFlag parameter was modified to support scroll wheels. A column was added to Table 3.3.24-2b to show direction of positive orientation for angles (clockwise or counter-clockwise).

3.3.25 Map_Horz

A new event was defined for Item Synchronization.

3.3.28 PicturePushButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.29 PictureToggleButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.30 PopUpPanel

A restriction was removed. PopUpPanels can be nested.

3.3.31 PopUpMenu

PictureArray was added as run-time modifiable parameter in Table 3.3.31-4.

3.3.32 PopUpMenuButton

A parameter was modified to include the identity of the widget to focus upon. The PictureArray parameter was added

3.3.33 PushButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.34 RadioBox

Restrictions were removed from this widget.

3.3.35 RotationContainer

The Enable parameter was deleted as it does not apply to this widget

3.3.36 ScrollPanel

The Horizontal Scroll parameter was modified to include Top/Bottom scroll. The Vertical Scroll parameter was modified to include Left/Right scroll.

3.3.37 ScrollList

The EnableArray parameter was added.

3.3.37.1 ScrollList Specific A661_ParameterStructure

New Section added.

3.3.39 TabbedPanel

A parameter was modified to include the identity of the widget to focus upon.

3.3.41 ToggleButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.42 TranslationContainer

The Enable parameter was deleted as it does not apply to this widget. Runtime parameters TranslationX and TranslationY were defined to be 8 bytes.

3.4.1 MapGrid

The description of MapGrid parameters, IncrementX and IncrementY, were modified in Table 3.4.1-1. The changes would distinguish between distance-distance on a rectangular grid versus true bearing-distance grid used with weather radar. Support for various MapDataFormat values would be implementation dependent.

3.4.3 MapVert

Vertical display capability is included for terrain profile and other applications. A new event was defined for Item Synchronization. This would ensure that weather radar display and terrain display would be synchronized with aircraft position. Parameters RangeX and RangeY were retained.

3.4.4 MapVert_Source

The MapDataFormat parameter described in Table 3.4.4-2b was modified to allow the origin of the map to be an absolute geometric point or a point relative to a geometric reference point. A clarification was made to the use of different MapDataFormats in this widget.

3.4.5 MapVert_ItemList

Item Synchronization was added to Table 3.4.5.1-1.

3.4.5.2.1.10 Item_Synchronization

New Section added.

3.4.5.2.11 Symbol_Rotated

New Section added.

3.4.5.3 MapVert_ItemList Interactive Item

New Section added.

3.4.6 EditBoxMultiline

Changes made to event reporting of this widget.

3.4.7 ComboBoxEdit

A parameter was modified to include the identity of the widget to focus upon. Changes made to event reporting and entry validation.

3.5 Widget Library Expansion

This section and its subordinate sections were added. Seven widgets are included in the widget library expansion in supplement 2.

3.5.1 MutuallyExclusiveContainer widget

MutuallyExclusiveContainer widget was added to activate a single widget from a collection of widgets.

3.5.2 ProxyButton widget

ProxyButton widget was added to direct a physical button push as an event selection on a window.

3.5.3 WatchdogContainer widget

WatchdogContainer widget was added to monitor the refresh rate of data supplied to the display system.

3.5.4 Slider widget

Slider widget was added to enable scrolling through the contents of a specific viewing area.

3.5.5 PictureAnimated widget

PictureAnimated widget was added to display of a set of bitmap images in rapid succession.

3.5.6 SymbolAnimated widget

SymbolAnimated widget was added to animate vector symbols.

3.5.7 SelectionListButton widget

SelectionListButton widget was added to allow a crew member to select one entry within a list.

4.5.1 Notation

Notation was corrected to specify [<A>] means zero or one of [<A>].

4.5.3.1 UADF loading structure

This section was added to describe the software data loading of the User Application Definition File (UADF).

4.5.3.2 Definition File (DF) Structure

The OEM File Header was replaced with the ARINC 665 definition file header, part of which can be implementation dependent. The block structure for Symbol Graphical Definition (SGD) was added.

4.5.3.5 Definition Time Symbol Block Commands

This section was added to address the symbol graphical definition (SGD) language.

4.5.3.6 Symbol Command Structure

This section added to address the symbol graphical definition (SGD) language.

4.5.3.7 Constraints Inside a Symbol Definition Block

This section was added to introduce Symbol Graphical Definition (SGD).

4.5.4.5.7 A661_Parameter_Structure_EnableArray

This section was added to support new widgets.

4.6 ARINC 661 Keyword Values

Section 4.6 was updated to include new constants, keywords and associated values.

5.0 Symbol Graphical Definition (SGD)

Symbol Graphical Definition (SGD) was added to define complex symbology.

6.0 XML Definition File Specification

This section was added as a suggested encoding method for ARINC 661 Definition Files.

Appendix C Example of a Definition File

This Appendix was updated to support the XML Definition File Specification added in Section 6. Section C.5 was added to provide an example XML definition file with a wide range of properties. Section C.6 was added as an example binary definition file that contains two symbol definitions. Section C.7 is the XML encoding of the binary definition file provided in Section C.6.

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7465

SUPPLEMENT 3

TO

ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACE TO USER SYSTEMS

Published: November 7, 2007

Prepared by the AEEC

Adopted by the Airlines Electronic Engineering Executive Committee: September 26, 2007

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to **ARINC Specification 661:** *Cockpit Display System Interface to User Systems*. The supplement introduces six new widgets that expand the capability of ARINC 661 display systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document **blue bold** text is used to indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

Global Changes to ARINC 661 Introduced by Supplement 3

For reasons of conciseness, global changes introduced by this supplement are summarized here. These changes are repetitive changes that affect multiple sections in the same way.

An improvement to the handshake protocol between the CDS and the UA was introduced for validating pilot entries to the CDS. As part of this UA Validation change, the Enable parameter was redefined to be a three-state enumerated data type instead of a Boolean flag. Widget Parameter Tables, Creation Structure Tables and Runtime-Modifiable Parameter Tables were modified accordingly.

Other global changes:

- Added Type column to several Event Structure Tables.
- Added Size column to several Runtime-Modifiable Parameter Tables.
- Removed Values column from the four Runtime-Modifiable Parameter Tables that had this column: the information is found in the Widget Parameter Tables.
- Added notes to several widget description sections to clarify cases in which the cursor collides with widgets or layers which overlap one another.

Numerous minor editorial changes were applied to the document to improve:

- Table and Figure numbering consistency
- Text and Table formatting consistency
- Clarity and Grammar of Text descriptions.

2.2.4 ARINC 661 Conformance

This section was updated to better describe ARINC 661 compliance.

A reference to Appendix G was added to describe how new widgets can be introduced in ARINC 661.

2.3.5.2 From CDS to UA

Added clarification describing the behavior of overlapping widgets.

3.1.2.1 Widget States Definition

The section was expanded by the inclusion of Figure 3.1.2 (formerly included in Section 3.1.2.2).

3.1.2.2 Inner State Management: “Race Condition”

This section was updated to describe mitigation of race conditions. The reference to Appendix D was removed. (Appendix D was deleted by Supplement 3.)

3.1.3.5 Parameters Related to Focus Navigation

This Section was modified to support FocusLink widget introduced in Section 3.6.4.

3.1.4 Widget Events

Widget Event Use Cross Reference Table 3.1.4-1 was added.

3.2.1 Widgets Summary

The widget library summary was expanded to include six new widgets introduced in Supplement 3. These include:

3.6.1 EditBoxNumericBCD
3.6.2 CursorRef
3.6.3 CursorOver
3.6.4 FocusLink, FocusIn, FocusOut
3.6.5 SizeToFitContainer
3.6.6 ShuffleToFitContainer

3.2.2 Widgets Classification

Table 3.2.2-1 and 3.2.2-2 was updated to support the widget expansion in Supplement 3. New widget categories “utility” and “UA validation” were added.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 was expanded to include new widgets and the relationship with respect to each other.

3.2.5.1 Available Character Set

Clarify character set.

3.2.8.4 Map Synchronization Number

New Section added.

3.2.9 UA Validation

This Section replaces the old Section 3.2.9 (Non-Classified Widgets) deleted by Supplement 3. UA Validation describes the handshake protocol between the CDS and the UA for validating pilot entries and selections.

3.3.4.1 Buffer Format Alignment

This section renamed. The content remains the same.

3.3.6 ComboBox

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete and what is displayed during entry validation.

3.3.9 EditBoxMasked

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.10 EditNumeric

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.11 EditBoxText

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.20 Label

Changed Font parameter so it is runtime-modifiable.

3.3.22 MapHorz_ItemList Parameters

The Map Item Synchronization number was added. Table 3.3.22-1 - MapHorz_ItemList Parameters was updated. Table 3.3.22-4 - MapHorz_ItemList Runtime Modifiable Parameters was updated. Made spelling of BufferOfMapItems ParameterIdent match Section 4.

3.3.22.1 MapHorz_ItemList Standard Items Description

New MapHorz_Items for Symbol Target and Triangle Strips/Fans were added to Table 3.3.22.1 - MapHorz_ItemList Standard Items Description.

3.3.22.2.1.15 Symbol_Target

New section added.

3.3.22.2.1.16 Triangle Strip Map Item

New section added.

3.3.22.2.1.17 Triangle Segment Map Item

New section added.

3.3.22.2.1.18 Triangle Segment Double Map Item

New section added.

3.3.22.2.1.19 Triangle End Map Item

New section added.

3.3.22.2.1.20 Triangle End Double Map Item

New section added.

3.3.22.2.1.21 Triangle Fan Start Map Item

New section added.

3.3.25 MapHorz Parameters

The Map Item Synchronization number was added. Table 3.3.25-1 - MapHorz Parameters was updated. Table 3.3.25-3 - MapHorz Runtime Modifiable Parameters was updated.

3.3.27 Panel

The motion allowed parameter was added. The position and size parameters are run-time modifiable. Table 3.3.27-1, Panel Parameters, was updated. Table 3.3.27-2, Panel Creation Structure, was updated. Table 3.3.27-3, Panel Runtime Modifiable Parameters, was updated.

3.3.37 ScrollList

This section updated for clarity. Two parameters were added.

3.4.1 MapGrid

The Map Item Synchronization number was added. Table 3.4.1-1 - MapGrid Parameters was updated. Table 3.4.1-3 - MapGrid Run-Time Modifiable Parameters was updated

3.4.3 MapVert

The Map Item Synchronization number was added. Table 3.4.3-1 - MapVert Parameters was updated. Table 3.4.3-3 - MapVert Runtime Modifiable Parameters was updated.

3.4.5 MapVert_ItemList

The Map Item Synchronization number was added. Table 3.4.5-1 - MapVert_ItemList Parameters was updated. Table 3.4.5-3 - MapVert_ItemList Runtime Modifiable Parameters was updated.

3.4.5.1 MapVert_ItemList Standard Items Description

New MapVert_Items for Triangle Strips and Fans were added to Table 3.4.5.1 - MapVert_ItemList Standard Items Description. Made spelling of BufferOfItems ParameterIdent match new spelling adopted in Section 4.

3.4.5.2.1.12 Triangle Strip MapVert_Item

This section was added.

3.4.5.2.1.13 Triangle Segment MapVert_Item

This section was added.

3.4.5.2.1.14 Triangle Segment Double MapVert_Item

This section was added.

3.4.5.2.1.15 Triangle End MapVert_Item

This section was added.

3.4.5.2.1.16 Triangle End Double MapVert_Item

This section was added.

3.4.5.2.1.17 Triangle Fan Start MapVert_Item

This section was added.

3.4.6 EditBoxMultiLine

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.4.7 ComboBoxEdit

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.5.6 SymbolAnimated

Renamed AnimationFlag and LoopFlag as AnimationType and LoopFlag, respectively, and changed tables to correctly describe these parameters as enumerated parameters.

3.5.7 SelectionListButton

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete and what is displayed during entry validation.

3.6 Widget Library Expansion

This section and its subordinate sections were added. Six widgets are included in the widget library expansion in Supplement 3.

3.6.1 EditBoxNumericBCD

This section was added to define a new widget.

3.6.2 CursorRef

This section was added to define a new widget.

3.6.3 CursorOver

This section was added to define a new widget.

3.6.4 Focus Navigation Widgets

This section and its subordinate sections were added to define three new widgets to allow focus motion between layers.

3.6.5 SizeToFitContainer

This section was added to define a new widget.

3.6.6 ShuffleToFitContainer

This section was added to define a new widget.

4.4.2 Error Notification

This section was updated to clarify CDS behavior in error conditions.

4.4.3.1 Request from UA to CDS

This section expanded to include the capability to set cursor on widget.

4.5.3.2 Definition File (DF) Structure

Added picture definitions. Table 4.5.3.2-2 was clarified.

4.5.4.3 Request Structure

Added Table 4.5.4.3-6 containing cursor on widget description.

4.5.4.5.9 A661_ParameterStructure_BufferOfItems

Added a reference and clarified that this structure is used for both MapHorz_ItemList and MapVert_ItemList.

4.5.4.5.10 A661_ParameterStructure_Buffer

Structure table was added.

4.6 ARINC 661 Keyword Values

This section was updated to include new constants, keywords and associated values.

Table 4.6-2 updated to include picture block.

Table 4.6-3B updated to include symbol size, symbol rectangular and symbol circular.

Table 4.6-5 updated to include request cursor on widget.

Table 4.6-7 updated to include new keyword values.

Table 4.6-8 updated to include new keyword values.

Table 4.6-9 updated to delete material pertaining to unused events.

Table 4.6-10 updated to include a True_With_Validation value for the redefined Enable parameter. This is part of the UA Validation change.

Table 4.6-11 updated to include new keyword values.

5.2 Symbol Definition Commands

Symbol definition commands were updated to define an optional sensitive area representation.

5.2.1.3 Sensitive Area

This section was added to describe cursor sensitive areas for symbols.

5.2.3.13 Size

This section was added to define a new symbol definition command which allows the size of a symbol representation to be set.

6.0 XML Definition File Specification

This section was expanded to support OEM extensions and guidance for representing non-printable characters.

Expanded to include sensitive area and picture graphical definition.

Expanded to include support for defining bitmap images in XML definition files.

7.0 Picture Graphical Definition

New section added. It describes how bitmap images can be described in binary definition files using data structures called picture blocks.

Appendix C - Example of Definition File

This Appendix was updated and expanded. Examples C-1 through C-7 were updated for clarity and to bring them in line with changes to widget parameters. Example C-8, Binary DF Example, was added. Example C-9, XML Example, was added. Both C-8 and C-9 are examples of using bitmaps in DFs.

Appendix D - Use of StyleSet for Inner State Depiction

This appendix was deleted by Supplement 3.

Appendix F - Communication Transport Protocols

This appendix was added by Supplement 3.

Appendix G - New Widget Guidelines

This appendix was added by Supplement 3.

ARINC Standard – Errata Report

1. Document Title

ARINC Specification 661-3: *Cockpit Display System Interfaces to User Systems*
Published: November 15, 2007

2. Reference

Page Number: _____ Section Number: _____ Date of Submission: _____

3. Error

(Reproduce the material in error, as it appears in the standard.)

4. Recommended Correction

(Reproduce the correction as it would appear in the corrected version of the material.)

5. Reason for Correction *(Optional)*

(State why the correction is necessary.)

6. Submitter (Optional)

(Name, organization, contact information, e.g., phone, email address.)

Please return comments to fax +1 410-266-2047 or standards@arinc.com

Note: Items 2-5 may be repeated for additional errata. All recommendations will be evaluated by the staff. Any substantive changes will require submission to the relevant subcommittee for incorporation into a subsequent Supplement.

[To be completed by IA Staff]

Errata Report Identifier: _____ **Engineer Assigned:** _____

Review Status: _____

ARINC IA Project Initiation/Modification (APIM)

1.0 Name of Proposed Project **APIM #: _____**

(Insert name of proposed project.)

2.0 Subcommittee Assignment and Project Support

2.1 Identify AEEC Group

(Identify an existing or new AEEC group.)

2.2 Support for the activity

Airlines: *(Identify each company by name.)*

Airframe Manufacturers:

Suppliers:

Others:

2.3 Commitment for resources *(Identify each company by name.)*

Airlines:

Airframe Manufacturers:

Suppliers:

Others:

2.4 Chairman: (Recommended name of Chairman.)

2.5 Recommended Coordination with other groups

(List other AEEC subcommittees or other groups.)

3.0 Project Scope *(why and when standard is needed)*

3.1 Description

(Insert description of the scope of the project. Use the following symbol to check yes or no below. ☒)

3.2 Planned usage of the envisioned specification

New aircraft developments planned to use this specification yes ❑ no ❑

Airbus: *(aircraft & date)*

Boeing: *(aircraft & date)*

Other: *(manufacturer, aircraft & date)*

Modification/retrofit requirement yes ❑ no ❑

Specify: *(aircraft & date)*

Needed for airframe manufacturer or airline project yes ❑ no ❑

Specify: *(aircraft & date)*

Mandate/regulatory requirement yes ❑ no ❑

Program and date: *(program & date)*

Is the activity defining/changing an infrastructure standard? yes ❑ no ❑

Specify *(e.g., ARINC 429)*

When is the ARINC standard required?
_____*(month/year)*_____

What is driving this date? _____*(state reason)*_____

Are 18 months (min) available for standardization work? yes ❑ no ❑

If NO please specify solution: _____

Are Patent(s) involved? yes ❑

If YES please describe, identify patent holder: _____

3.3 Issues to be worked

(Describe the major issues to be addressed.)

4.0 Benefits

4.1 Basic benefits

Operational enhancements yes ❑ no ❑

For equipment standards:

a. Is this a hardware characteristic? yes ❑ no ❑

b. Is this a softwareware characteristic? yes ❑ no ❑

c. Interchangeable interface definition? yes ❑ no ❑

d. Interchangeable function definition? yes ❑ no ❑

If not fully interchangeable, please explain: _____

Is this a software interface and protocol standard*?* yes ❑ no ❑

Specify: _____

Product offered by more than one supplier yes ❑ no ❑

Identify: *(company name)*

4.2 Specific project benefits

(Describe overall project benefits.)

4.2.1 Benefits for Airlines

(Describe any benefits unique to the airline point of view.)

4.2.2 Benefits for Airframe Manufacturers

(Describe any benefits unique to the airframe manufacturer's point of view.)

4.2.3 Benefits for Avionics Equipment Suppliers

(Describe any benefit unique to the equipment supplier's point of view.)

5.0 Documents to be Produced and Date of Expected Result

5.1 Meetings and Expected Document Completion

The following table identifies the number of meetings and proposed meeting days needed to produce the documents described above.

Activity	Mtgs	Mtg-Days (Total)	Expected Start Date	Expected Completion Date
Document a	*# of mtgs*	*# of mtg days*	*mm/yyyy*	*mm/yyyy*
	*# of mtgs **	*# of mtg days **		
Document b	*# of mtgs*	*# of mtg days*	*mm/yyyy*	*mm/yyyy*
	*# of mtgs **	*# of mtg days **		

** Indicate unsupported meetings and meeting days, i.e., technical working group or other ad hoc meetings that do not requiring IA staff support.*

6.0 Comments

(Insert any other information deemed useful to the committee for managing this work.)

For IA Staff use

Date Received: _____ IA Staff Assigned: _____

Estimated Cost: _____

Potential impact: _____

*(**A**. Safety **B**. Regulatory **C**. New aircraft/system **D**. Other)*

Forward to committee(s) (AEEC, AMC, FSEMC): _____ Date Forwarded:_____

Committee resolution: _____

*(**0** Withdrawn **1** Authorized **2** Deferred **3** More detail needed **4** Rejected)*

Assigned Priority:_____ Date of Resolution:_____

*(**A** High - execute first **B** Normal - may be deferred.)*

Assigned to SC/WG:_____