ARINC

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

ARINC SPECIFICATION 661-1

PUBLISHED: JUNE 26, 2003

AN ARINC DOCUMENT

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

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. Any use of or reliance on this document shall constitute an acceptance thereof “as is” and be subject to this disclaimer.

Copyright© 2003 by
AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 21401-7465 USA

ARINC SPECIFICATION 661-1©

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

Published: June 26, 2003

Prepared by the Airlines Electronic Engineering Committee

Specification 661	Adopted by the Airlines Electronic Engineering Committee:	November 12, 2001
Specification 661	Air Industry Review Completed:	December 28, 2001

Summary of Document Supplements

Supplement	Adoption Date	Published
Specification 661-1	March 6, 2003	June 26, 2003

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. Today, the major U.S. airlines remain the Company's principal shareholders. Other shareholders include a number of non-U.S. airlines and other aircraft operators.

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 Airlines Electronic Engineering Committee (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:

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

b) 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.

ARINC REPORT 661
TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose and Scope

The purpose of this document is to define interfaces to a Cockpit Display System (CDS) used in all types of aircraft installations. The primary objective is to minimize the cost to the airlines, directly or indirectly by accomplishing the following:

a. Minimize the cost of acquiring new avionic systems to the extent it is driven by the cost of CDS development.

b. Minimize the cost of adding new display function to the cockpit during the life of an aircraft.

c. Minimize the cost of managing hardware obsolescence in an area of rapidly evolving technology.

d. 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 interface 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 that is 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 Characteristics, such as:

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

1.0 INTRODUCTION

1.3 Interoperability

1.3.1 General

One of the primary objectives of this Specification 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 use 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:

1.0 INTRODUCTION

1.9 Applicability (cont'd)

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)

2.0 CONCEPT OF OPERATION

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 by way of 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 they use in the initial design or are expected 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 their 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.

2.0 CONCEPT OF OPERATION

2.2 Overview of Interface Level Between UA and CDS (cont'd)

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.

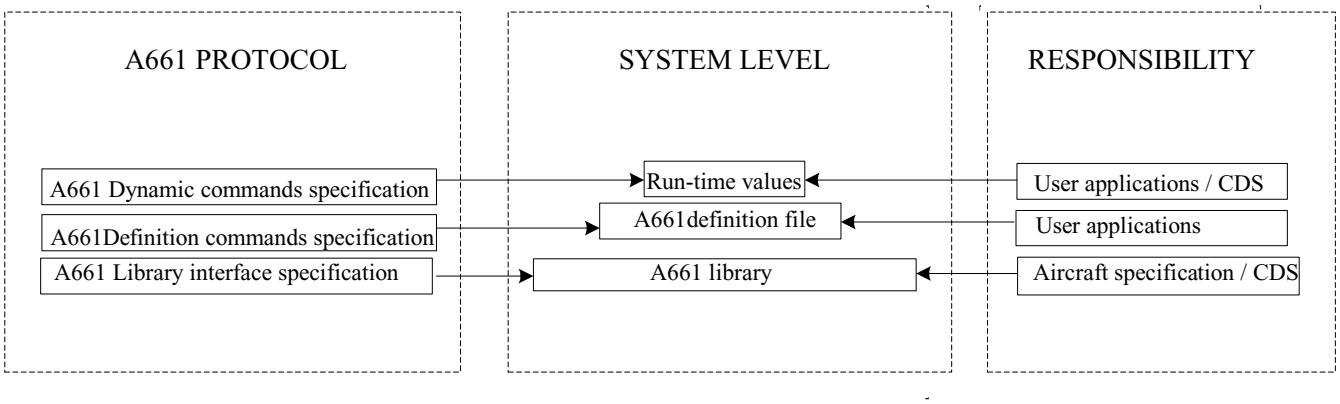


Figure 2.1-1 ARINC 661 Protocol Principles

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. c-1

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 one DF before the beginning of the run-time data exchange for widgets defined inside this DF (run-time phase for this DF).

2.0 CONCEPT OF OPERATION

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:

Update the run-time widget parameters.

Request to the CDS for change 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 command, 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 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

c-1

In some conditions, the CDS may loose 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 this standard when it implements all ARINC 661 mechanisms, all standardized widgets and potentially other widgets, as necessary.

A UA conforms to this standard when it implements the necessary ARINC 661 mechanisms and uses only the standardized widgets.

2.0 CONCEPT OF OPERATION

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.

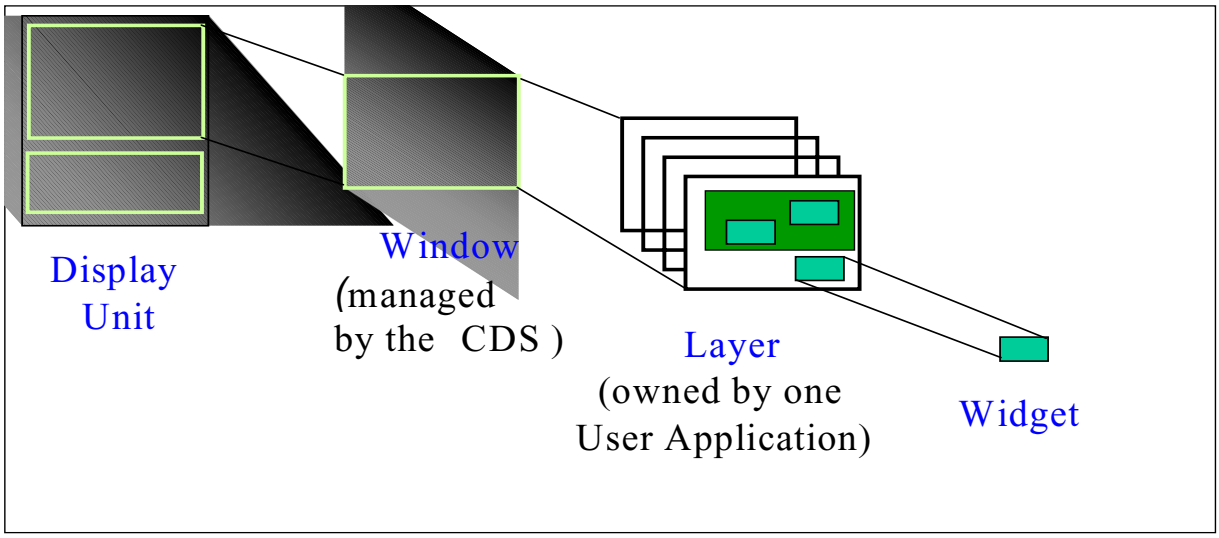


Figure 2.3-1 Window/Layer Illustration

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.

2.0 CONCEPT OF OPERATION

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 one graphical layer associated with one 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 UA 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 requires:

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:

One ARINC 661 Layer is associated with one 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 the knowledge of its layer, not the knowledge of the containing window, which is defined by current CDS configuration.

2.3.2.2 Layer Content Management (cont'd)

One UA or the CDS itself, possibly, owns several layers within a window.

The owner of a layer is responsible for managing the parameters of the contained widget. In this way, 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 to 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 data of its layer

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.

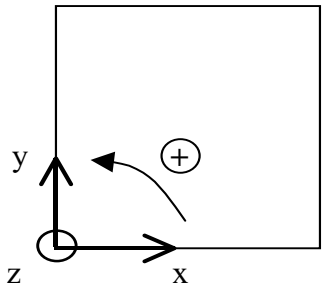


Figure 2.3.4
Graphic references for widget positioning

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.

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.

2.3.5.1 From UA to CDS

An example of cursor management from the UA to the CDS is to request the highlight of 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

Determination of all cases where a 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.

2.0 CONCEPT OF OPERATION

COMMENTARY (cont'd)

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

System integrators and designers are reminded that the flight deck is not a 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.

3.0 WIDGET LIBRARY

3.1 Introduction to Widgets

Communication between the CDS and UA is defined based on the identification of widgets defined in Section 3.0, 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 UALD.

Since the CDS also 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, [UserApplicaionIdent] 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, have to be considered to define widget states:

1. Visibility level: widget is visible or not

2. 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"

3. 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.

4. 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 one 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

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:

Upon selection by a crew member, the CDS will change the widget inner state from SELECTED to UNSELECTED or from UNSELECTED to SELECTED.

The UA may change the inner state to initialize or refresh the interface reasons.

Therefore, a conflict situation may occur that is based on the fact that inner states of an interactive widget are generally managed by the CDS upon crew member interaction. When the CDS changes the state of a widget, it sends an event to inform the UA of the interaction. In this case, the widget is considered as an IN widget. Interactions from the crew member enter the IN widget.

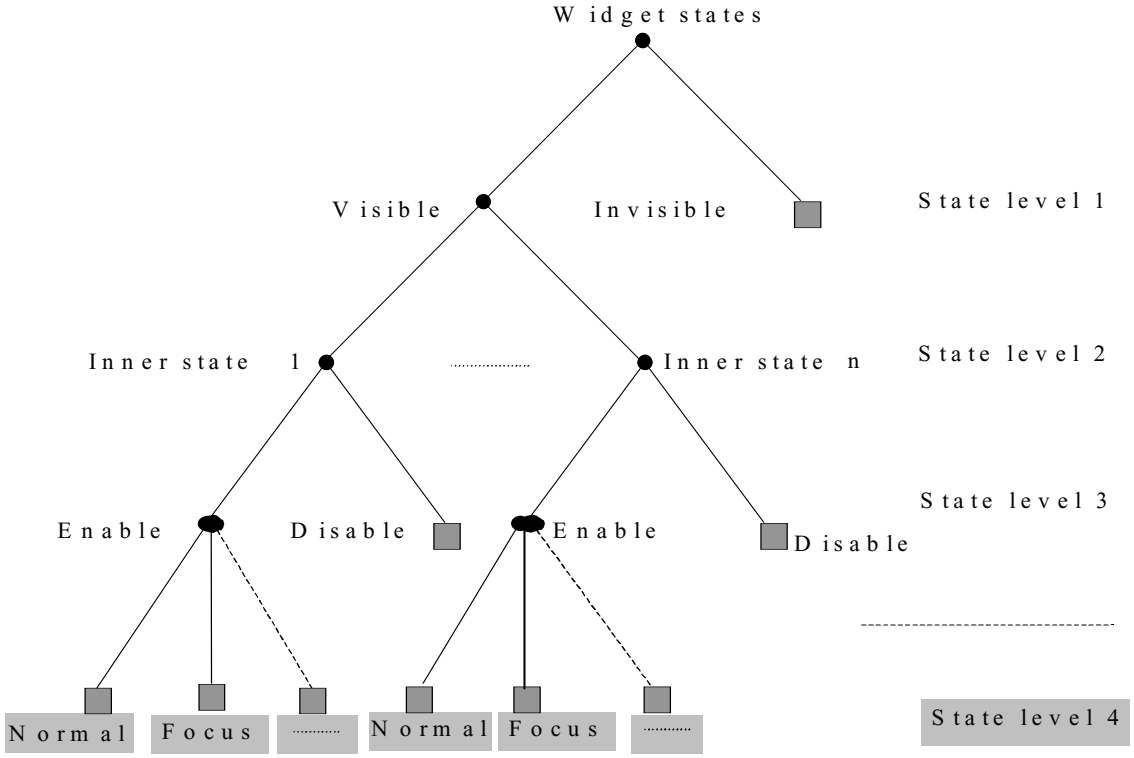


Figure 3.1.2 Example of Widget States Levels

3.0 WIDGET LIBRARY

3.1.2.2 Inner State Management: "Race Condition" (cont'd)

But the inner state is not supposed to reflect a functional context of the UA. This means that the inner state should not be used to display the fact that the UA has taken into account the interaction. If the UA wants to display its functional context though the interactive widget, the widget is considered as an OUT widget. The widget is used to provide functional information to the crew member. This information hould be covered by "StyleSet" parameter.

The CDS has the responsibility to provide a graphical feedback upon a crew member interaction through the inner state. The UA has the responsibility to provide a functional feedback on crewmember interaction. If the UA wants to display this functional feedback on the widget itself, it should use the "StyleSet" parameter.

An illustration of the management of an IN/OUT widget is provided in Appendix D, Example of "IN/OUT" Widget Management Using "StyleSet" parameter.

If inner states are used for both command and functional context, their management by both CDS and UA will lead to conflicts known as a "race condition."
However, the UA will have to change inner states of its widgets, for initialization or refresh of its interface. The UA should take into account the probability of "race condition" for example, turning off and then on the visibility (or interactivity) of the widgets after refresh of their inner states.

Different types of race conditions have been identified for managing an inner state, for example:

a. Command from a UA arrived just before an interaction from a crew member in the CDS.

   The CDS must take into account the UA command, where the UA is the master of widget management.

   COMMENTARY

   The CDS behavior on the event reception is not in the scope of this Specification. It depends on the CDS implementation.

b. An interaction from a crew member arrived just before the command from a UA. The CDS will send an event to the UA and then take into account the UA command. The UA will receive an event with the context number attached. If the UA has changed the context number through the command it sent, the context number received allows the UA to know that the event has been generated before the reception of its command.

3.0 WIDGET LIBRARY

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 the widget
WidgetIdent	Identifier of the widget (refer to the Section 3.1.1, 3.1.1 Widget Identification) 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 the type of the widget.
Visible	A661_FALSE: The widget will not be rendered. A661_TRUE: a. If all its parent are visible, the widget will be rendered. b. If one of all its parents is 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: a. If all its parent are enabled, the widget will be interactive. b. If one of all its parents is disabled, the widget will not be interactive, whatever the value of its Enable parameter. COMMENTARY 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. The CDS behavior when a UA attempts to SetParameter on an anonymous widget is undefined.

3.0 WIDGET LIBRARY

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 user application 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. Requiring or disallowing any of these characteristics is incompatible with the spirit of this standard. 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 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. The aircraft OEM (or CDS vendor) defines the list of StyleSet values. COMMENTARY Examples of possible StyleSet values: STYLE_SET_NOMINAL STYLE_SET_ADVISORY STYLE_SET_CAUTION STYLE_SET_WARNING STYLE_SET_SELECTED STYLE_SET_PRESELECTED STYLE_SET_ENGAGED 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 widget 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

The management of directional motion of the focus, e.g., through arrow keys, will be internal to CDS and as a consequence, 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 “FocusIndex” 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 “FocusIndex” parameter.

The Widget Common Structure is defined 3.1.3.5.

Table 3.1.3.5 - Widget Common Structure

Parameter	Description
FocusIndex	Order of the widget (FocusIndex is specified from “1” to “n” that it applies to one layer, FocusIndex “n” next is “1”). When multiple layers are present, the behavior will be specified by OEM. If set to 0, the widget is excluded from the focus motion list.
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 FocusIndex parameter

3.2 HMI Widget Library Summary

3.2.1 Widgets Summary

Table 3.2.1 summarizes the widgets in the Widget Library.

3.0 WIDGET LIBRARY

Table 3.2.1 - Widgets Library Summary

Widget Type	Description
1 ActiveArea	Transparent rectangular area defining an interactive area. Selection of this area will send an event to the owner application.
2 BasicContainer	Manages the visibility and the interactivity of a group of widgets.
3 BlinkingContainer	Allows to apply blinking behavior to a group of widgets.
4 BufferFormat	The objective of this widget is to provide a mean 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.
5 CheckButton	A CheckButton allows the crew-member to select or not an option. (other names: Radio/Toggle box).
6 ComboBox	A ComboBox is a widget providing a mean 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.
7 Connector	The purpose of this widget is to connect a layer to a container of another layer. In this way it provides the mean to 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. c-1
8 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.
9 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.”
10 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.
11 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).
12 GpArcEllipse	The graphical primitive GpArcEllipse allows the definition of an arc (portion of an ellipse or a circle).
13 GpArcCircle	The graphical primitive GpArcCircle allows the definition of a circular arc.
14 GpCrown	The graphical primitive GpCrown allows the definition of a circular filled region.
15 GpLine	The graphical primitive GpLine allows the definition of a line.
16 GpLinePolar	The graphical primitive GpLinePolar allows the definition of a line using a polar definition.
17 GpRectangle	The graphical primitive GpRectangle allows the definition of a rectangle.
18 GpTriangle	The graphical primitive GpTriangle allows the definition of a triangle.
19 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 color not modifiable (unlike characters). Picture has no rotation capability.
20 Label	A Label consists of a non-editable text field at a defined display location.

3.0 WIDGET LIBRARY

Widget Type	Description
21 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.
c-1 22 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.
23 MapLegacy	Map Legacy widget provides a mean for being compatible with currently use data format, such as ARINC 702 format for FMS, AEEC 453 format for weather radar.
c-1 24 MapHorz_Source	MapHorz_Source is a specialized container. It contains a specific dynamic widget expressed in a common coordinate system. It describes characteristics of the common coordinate system.
c-1 25 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.
26 MaskContainer	MaskContainer is intended to apply a referenced mask to a group of widgets.
27 Panel	A panel groups several widgets together in a rectangular area and has clipping capabilities.
28 PicturePushButton	Momentary switched button with Picture, which allows the crew-member to launch an action (to send an event to the owner user application).
29 PictureToggleButton	Two stable states button with Picture.
30 PopUpPanel	PopUpPanel is a container that is displayed on the top of other layers. PopUpPanel visibility can be managed by the CDS using logic defined by the OEM. In this way this PopUpPanel can not be used as a regular container.
31 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.
32 PopUpMenuButton	PopUpMenuButton is a button providing the ability to display a PopUpMenu. This mechanism is internal to the CDS.
33 PushButton	Momentary switched button, which allows the crew-member to launch an action (to send an event to the owner user application).
34 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.
35 RotationContainer	A RotationContainer has all of the capabilities of Panel. Its intended use is to apply a rotation transformation to a group of widgets.
36 ScrollPanel	A ScrollPanel is a “sheet container widget”, for which only a subpart is visible called the “frame”. Scrolling button provide the capability to scroll the visible part inside the whole sheet.
37 ScrollList	A ScrollList is a list of items, for which only a subpart is visible. Scrolling buttons provide the capability to scroll the visible part of items inside the whole list.
38 Symbol	Symbol has rotation and color capability. Symbol widget has a reference to a table.
39 TabbedPanel	A TabbedPanel widget is a Panel associated with a selection button. This widget is only for use within a TabbedPanelGroup widget.
40 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.

3.0 WIDGET LIBRARY

Widget Type	Description
41 ToggleButton	Two stable states button with text.
42 TranslationContainer	A TranslationContainer is similar to a Panel. Its intended use is to apply a translation transformation to a group of widgets.

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.

Table 3.2.2-2 - Widget Classification Table

	Widget Categories/Widgets	Container	Map Manage-ment	Dynamic Motion	Graphical Represen-tation	Text string	Interactive
3.3.1	ActiveArea				X		X
3.3.2	BasicContainer	X					
3.3.3	BlinkingContainer	X					
3.3.4	BufferFormat						
3.3.5	CheckButton				X	X	X
3.3.6	ComboBox				X	X	X
3.3.7	Connector						
3.3.8	CursorPosOverlay						X
3.3.9	EditBoxMasked				X	X	X
3.3.10	EditBoxNumeric				X	X	X
3.3.11	EditBoxText				X	X	X
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		

3.0 WIDGET LIBRARY

	Widget Categories/Widgets	Container	Map Manage-ment	Dynamic Motion	Graphical Represen-tation	Text string	Interactive
3.3.19	Picture				X		
3.3.20	Label			X	X	X	
3.3.21	LabelComplex				X	X	
c-1 3.3.22	MapHorz_ItemList		X		X	X	X
3.3.23	MapLegacy		X		X		
c-1 3.3.24	MapHorz_Source	X	X				X
3.3.25	MapHorz	X	X				
3.3.26	MaskContainer	X					
3.3.27	Panel	X			X		
3.3.28	PicturePushButton				X	X	X
3.3.29	PictureToggleButton				X	X	X
3.3.30	PopUpPanel	X			X		X
3.3.31	PopUpMenu				X	X	X
3.3.32	PopUpMenuButton				X	X	X
3.3.33	PushButton				X	X	X
3.3.34	RadioBox	X					
3.3.35	RotationContainer	X					
3.3.36	ScrollPanel	X			X		X
3.3.37	ScrollList				X	X	X
3.3.38	Symbol			X	X		
3.3.39	TabbedPanel	X			X	X	
3.3.40	TabbedPanelGroup	X			X		X
3.3.41	ToggleButton				X	X	X
3.3.42	TranslationContainer	X					
	WIDGET EXPANSION SUPPLEMENT 1						
3.4.1	MapGrid		X		X		
3.4.2	ExternalSource	X					
3.4.3	MapVert	X	X				
c-1 3.4.4	MapVert_Source	X	X				X
3.4.5	MapVert_ItemList		X		X	X	X
3.4.6	EditBoxMultiLine				X	X	X
3.4.7	ComboBoxEdit				X	X	X
3.4.8	MenuBar	X					X

3.0 WIDGET LIBRARY

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 can not 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 the layer is not a widget, it has been listed with the Container 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	Panel	PopUpPanel	RadioBox	RotationContainer	ScrollPanel	TabbedPanel	TabbedPanelGroup	TranslationContainer
ActiveArea	X		X							X	X			X	X		
BasicContainer	X		X							X	X			X	X		
BlinkingContainer	X		X					X		X	X		X	X	X		X
BufferFormat			X														
CheckButton	X		X							X	X	X		X	X		
ComboBox	X		X							X	X			X	X		
Connector				X		X				X					X	X	
CursorPosOverlay	X		X							X	X			X	X		
EditBoxMasked	X		X							X	X			X	X		
EditBoxNumeric	X		X							X	X			X	X		
EditBoxText	X		X							X	X			X	X		
GpArcCircle	X	X	X					X		X	X		X	X	X		X
GpArcEllipse	X	X	X					X		X	X		X	X	X		X
GpCrown	X	X	X					X		X	X		X	X	X		X
GpLine	X	X	X					X		X	X		X	X	X		X
GpLinePolar	X	X	X					X		X	X		X	X	X		X
GpRectangle	X	X	X					X		X	X		X	X	X		X
GpTriangle	X	X	X					X		X	X		X	X	X		X
Label	X	X	X					X		X	X		X	X	X		X
LabelComplex	X		X							X	X			X	X		
MapHorz	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		
Panel	X		X							X	X			X	X		
Picture	X		X							X	X			X	X		
PicturePushButton	X		X						X	X	X			X	X		
PictureToggleButton	X		X							X	X	X		X	X		
PopUpMenu	X		X							X	X			X	X		
PopUpMenuButton	X		X						X	X	X			X	X		
PopUpPanel	X		X							X				X	X		
PushButton	X		X						X	X	X			X	X		
RadioBox	X		X							X	X			X	X		
RotationContainer	X	X	X					X		X	X		X	X	X		X
ScrollList	X		X							X	X			X	X		
ScrollPanel	X		X							X	X			X	X		
Symbol	X	X	X					X		X	X		X	X	X		X
TabbedPanel			X													X	
TabbedPanelGroup	X		X							X	X			X	X		
ToggleButton	X		X							X	X	X		X	X		

c-1

Parents / Children	BasicContainer	BlinkingContainer	Layer	MapHorz	MapHorz_Source	MapVert	MapVert_Source	MaskContainer	MenuBar	Panel	PopUpPanel	RadioBox	RotationContainer	ScrollPanel	TabbedPanel	TabbedPanelGroup	TranslationContainer
TranslationContainer	X	X	X					X		X	X		X	X	X		X
WIDGET EXPANSION SUPPLEMENT 1																	
ComboBoxEdit	X		X							X	X			X	X		
EditBoxMultiLine	X		X							X	X			X	X		
ExternalSource	X		X					X		X			X		X		X
MapGrid					X		X										
MapVert	X		X							X	X			X	X		
MapVert_ItemList							X										
MapVert_Source			X			X		X									
MenuBar	X		X							X	X				X		

c-1

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 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:

ComplexLabel

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 the “MaxStringLength” parameter.

3.2.5.1 Available Character Set

Table 3.2.5.1 defines the characters available for text strings inside widgets. The characters in this table are representative of the shapes. It is not intended to define a font.

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’, ‘0a’, ‘0c’}

3.2.5.3 Change Style Capabilities

Table 3.2.5.3-1 lists the available change style capabilities through escape sequences.

Table 3.2.5.3-1 - Available Character Set

Control characters

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.

3.0 WIDGET LIBRARY

Table 3.2.5.3-1 - Available Character Set (cont'd)

Printing characters (ASCII)

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

Printing characters (A661 Extensions)

h80	△	overfly triangle
h81	°	degrees
h82	◇	diamond
h83	□	box

h84	←	Left arrow
h85	→	Right arrow
h86	↑	Up arrow
h87	↓	Down arrow

3.2.5.3 Change Style Capabilities (cont'd)

Table 3.2.5.3 lists the Escape Sequence Types.

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.
c-1 Animation	TAnimation	Animation of ACSII 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.
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 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 3.2.5.4, 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 h41) 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.

3.0 WIDGET LIBRARY

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

c-1

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
parameter = {‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’’, ‘6’, ‘7’, ‘8’, ‘9’, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’}

Notation: T line on the Top
L line on the Left
R line on the Right
B line on the Bottom

The text characters correspond to the following situation:

‘0’ = h30 = no line
‘1’ = h31 = B
‘2’ = h32 = T
‘3’ = h33 = B+T
‘4’ = h34 = L
‘5’ = h35 = B+R
‘6’ = h36 = B+L
‘7’ = h37 = T+L
‘8’ = h38 = R
‘9’ = h39 = T+R
‘A’ = h3A = R+L
‘B’ = h3B = B+L+R
‘C’ = h3C = L+T+R
‘D’ = h3D = B+R+T
‘E’ = h3E = L+B+T
‘F’ = h3F = B+T+L+R (Frame)

3.2.5.5 Escape Sequences Description (cont'd)

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
KbackGround	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

c-1

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, 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:

c-1

For Horizontal Map:

- MapHorz
- MapHorz_Source
- MapHorz_ItemList
- MapGrid

3.0 WIDGET LIBRARY

For Vertical Map:

- MapVert
- MapVert_Source
- MapVert_ItemList
- MapGrid

For more detailed information about 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 were part of avionics systems for quite a while. On the other hand the vertical map is slowly beginning to make an appearance. For this reason more in dept analysis is performed for the Horizontal Map.

3.2.8.1 Horizontal Map Management

A typical example of horizontal map management widgets is 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. c-1

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 allows to draw 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 MapHort_Source. Data in MapHorz_ItemList is meaningless without MapHort_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.

3.2.8.1 Horizontal Map Management (cont'd)

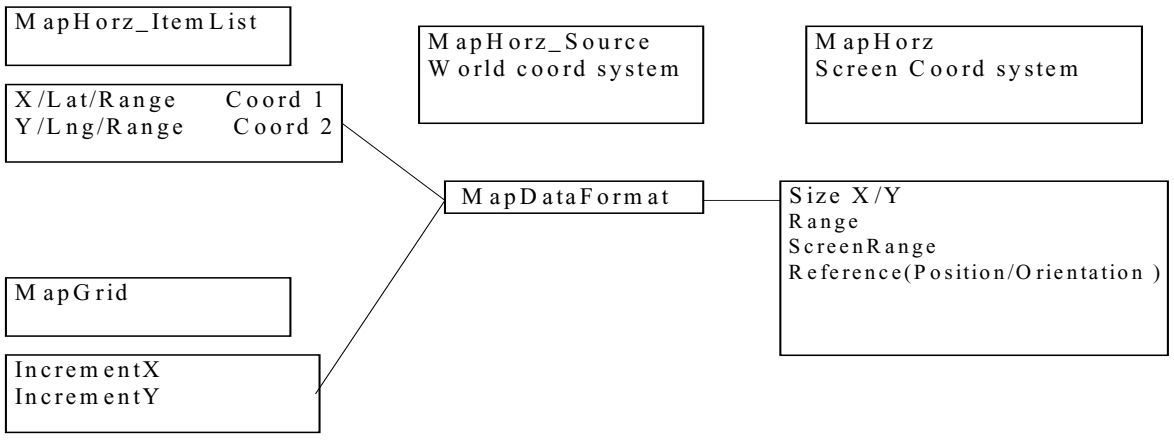


Figure 3.2.8.1 Coordinate System for MapHorz Widget Management

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.

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

c-1

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 Specification.

3.2.8.1.1 Link Between MapHorz, MapHorz_Source, MapHorz_ItemList and MapGrid

c-1

From a hierarchical point of view, illustrated in Figure 3.2.8.2, the 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.

3.0 WIDGET LIBRARY

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. c-1

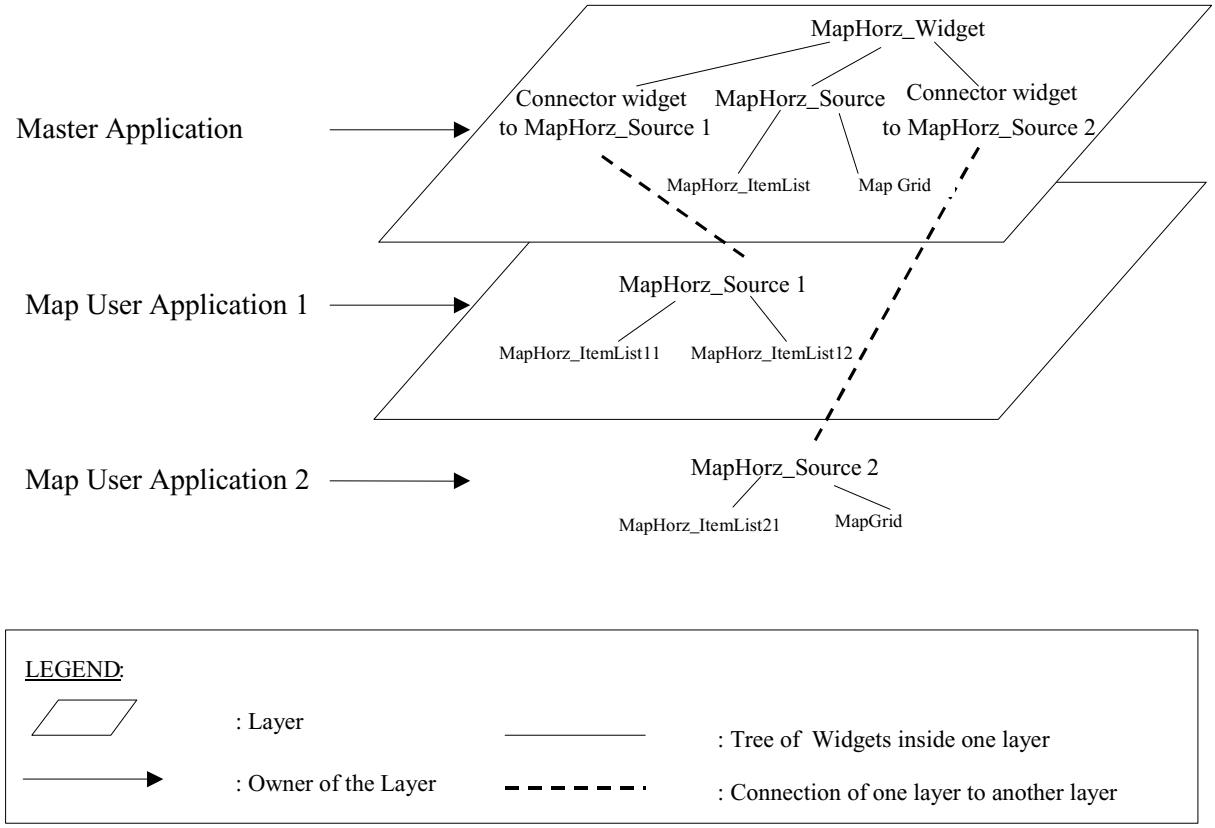


Figure 3.2.8.1.1 Hierarchical Structure For MapHorz Widget Management

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. c-1

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.

3.2.8.1.2 Parameter Definition for MapHorz and MapHorz_Source cont'd)

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 the MAPHORZ X,Y
- True North Angle :	Angle between the 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

c-1 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.

3.0 WIDGET LIBRARY

COMMENTARY

The FMS could set in different MapHorz_ItemLists the different flight plans, the different kinds of background data, etc. The MapHorz_ItemList allows the FMS to group items with different graphical priorities, which correspond to different functional group. c-1

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

3.2.9 Non-Classified Widget

Some widgets have been defined without graphical representation neither interactive or container capability. This widget has specific functionality in order to extend or optimize the ARINC 661 defined principle:

BufferFormat
Connector

3.3 Widget List

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

1. Definition section
2. Widget parameters table
3. Creation structure table: CreateParameterBuffer
4. Event Structure table
5. Run-time modifiable parameter tables
6. Specific sections

These parts are described as follows:

Part 1) This subsection states the categories of the widget, the functional description of this widget and some restrictions to the ARINC 661 principles.

Part 2) This subsection 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

3.0 WIDGET LIBRARY

3.3 Widget List (cont'd)

The next sections describe the format of the exchanges at definition-time, the Create command, as well as at run-time, the Widget Event notifications and SetParameter commands for each widget. This description is completed by the fourth subsection.

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 8.381903175442e-8. fr(32768) in 32 bits is LSB 0.000030517578125.
N/A	Non Applicable

All signed numbers are two's complement form.

Part 3) This subsection 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.

3.0 WIDGET LIBRARY

Part 4) This subsection 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.

Part 5) This subsection 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 subsections 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 highlight 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.

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value

3.0 WIDGET LIBRARY

3.3.1 ActiveArea (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	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

c-1

3.0 WIDGET LIBRARY

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

c-1

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

c-1

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	

c-1

The BasicContainer widget does not send any event.

3.3.2 BasicContainer (cont'd)

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 X2	32 x 2	A661_POS_XY	A661_ParameterStructure_XY
PosX	long	32	A661_POS_X	A661_ParameterStructure_X
PosY	long	32	A661_POS_Y	A661_ParameterStructure_Y

c-1

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.

3.3.4 BufferFormat (cont'd)

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 structure.
  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 structure.
  A661_ParameterStructure_1Byte
  A661_ParameterStructure_2Bytes
  A661_ParameterStructure_4Bytes
  A661_ParameterStructure_String
  A661_ParameterStructure_XY

Variable-size structures can not 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 one byte 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. The maximum size of this buffer is the sum of the maximum size of each parameter

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 A661_ParameterStructure_Buffer

Eight-byte, 4-byte, and 2-byte parameters must not cross 32-bit boundaries.

The BufferFormat structure must be padded out to a multiple of 32-bits.

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

CheckButtom 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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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

c-1

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	
Visible	uchar	8	A661_FALSE A661_TRUE	
PosX	long	32		
PosY	long	32		
SizeX	ulong	32		
SizeY	ulong	32		
StyleSet	ushort	16		
FocusIndex	ushort	16		
MaxStringLength	ushort	16		
CheckButtonState	uchar	8	A661_SELECTED A661_UNSELECTED	
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT	c-1
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE	
PicturePosition	uchar	8	A661_LEFT A661_RIGHT	c-1
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
UnusedPad	N/A	8	0
CheckButtonState	uchar	8	A661_SELECTED A661_UNSELECTED

3.3.5 CheckButton (cont'd)

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	ARINC661_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

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 clipping area of its containing window.

COMMENTARY

The data/formatting displayed for current-item-selected during crew selection and UA validation is part of CDS internal behavior, and is beyond the scope of this document. For instance, the CDS could implement a graphical representation indicating that one text has been selected but not yet validated by the UA. In this case the UA could show that it has validated the selection through a SetParameter command on the SelectedEntry parameter.

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)
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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 lower than MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list.
EntryList [MaxEntryNumber]	DR	String array holding the list of entries.

c-1

3.0 WIDGET LIBRARY

3.3.6 ComboBox (cont'd)

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
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	
FocusIndex	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	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	8	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

c-1

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

3.0 WIDGET LIBRARY

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

ComboBox Runtime Modifiable Parameters 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	ARINC661_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

3.3.7 Connector

Categories:
None

Description:
The purpose this widget is to connect a layer to a container of another 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 still handled by the owning application. | c-1

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.

3.3.7 Connector (cont'd)

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.
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
UnusedPad	N/A	24	0

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

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
UnusedPad	N/A	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	

3.3.8 CursorPosOverlay (cont'd)

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, the masks for this character are 1 for numeric mask and 0 for alpha mask.
If a character is only alphabetic, 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

The data/formatting displayed for current-item-selected during crew editing and UA validation is part of CDS internal behavior, and is beyond the scope of this document. For instance, the CDS could implement a graphical representation indicating that one text has been edited but not yet validated by the UA. In this case the UA could show that it has validated the entry through a SetParameter command on either the LabelString or EditBoxState parameter, according to defined widget behavior. Similarly, the UA should display an ERROR mode through the use of the StyleSet parameter.

3.0 WIDGET LIBRARY

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 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. c-1

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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	TRUE CDS will report each update from the crew member while in edit mode (A661_EVT_VALUE_CHANGE) CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_VALUE_CHANGE_ABORTED) FALSE CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) Note: In all cases the CDS will report the entry in edit mode
AlphaMask	DR	Mask for Alpha character
NumericMask	DR	Mask for Numeric character

c-1

3.3.9 EditBoxMasked (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	ushort	16	
AlphaMask	ulong	32	
NumericMask	ulong	32	
StartCursorPos	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_FALSE A661_TRUE
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

Available SetParameter identifiers and associated data structure are defined in Table 3.3.9-6.

Table 3.3.9-6 - 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)	Value	
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE	
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE	c-1
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		

3.0 WIDGET LIBRARY

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.

COMMENTARY

The data/formatting displayed for current-item-selected during crew editing and UA validation is part of CDS internal behavior, and is beyond the scope of this document. For instance, the CDS could implement a graphical representation indicating that one text has been edited, but not yet validated by the UA. In this case, the UA could show that it has validated the entry through a SetParameter command on either the LabelString or EditBoxState parameter, according to defined widget behavior. Similarly, the UA should display an ERROR mode through the use of the StyleSet parameter.

c-1 When the EditBoxNumeric 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 may be set by the UA through the “ReportAllChanges” parameter. If ReportAllChanges is True and, after having entered a value, the crewmember finally aborts the edit, the CDS should send a specific event to the UA with the former validated Value as parameter of the event.

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	
FocusIndex	D	Order of the widget for focus circulation	
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.	
Specific parameters			
Value	DR	Value displayed by the edit box in normal mode.	
FormatString	D	String describing the format of the numeric field.	
Tics coarse	D	Coarse increment step for modification of the value with main wheel.	c-1
Tics fine	D	Fine increment step for modification of the value with secondary wheel.	
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	TRUE CDS will report each update from the crew member while in edit mode (A661_EVT_VALUE_CHANGE) CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_VALUE_CHANGE_ABORTED) FALSE CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) Note: In all cases the CDS will report the entry in edit mode	c-1
NumericKeyFlag	D	Ability to change the value with the numerical key TRUE FALSE	
MinValue	D	Minimum value of the float	
MaxValue	D	Maximum value of the float	
CyclicFlag	D	Possibility for cyclic modification of the value TRUE FALSE	

3.3.10 EditBoxNumeric (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	ushort	16	
Value	float	32	
Tics coarse	float	32	
Tics fine	float	32	
c-1 MinValue	float	32	
MaxValue	float	32	
StartCursorPos	ushort	8	
UnusedPad	N/A	24	0
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_FALSE A661_TRUE
c-1 Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
UnusedPad	N/A	8	
NumericKeyFlag	uchar	8	0
CyclicFlag	uchar	8	
UnusedPad	N/A	16	
FormatString	string	8 * string length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits

EditBoxNumeric Event Structures: A661_EVT_VALUE_CHANGE_ABORTED are defined in Table 3.3.10-3.

Table 3.3.10-3 - EditBoxNumeric Event Structures: A661_EVT_VALUE_CHANGE_ABORTED

c-1 EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_VALUE_CHANGE_ABORTED
UnusedPad	N/A	16	
Value	float	32	

EditBoxNumeric Event Structures: A661_EVT_VALUE_CHANGE are defined in Table 3.3.10-4.

Table 3.3.10-4 - EditBoxNumeric Event Structures: 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	

EditBoxNumeric Event Structures: A661_EVT_VALUE_CONFIRMED are defined in Table 3.3.10-5.

Table 3.3.10-5 - EditBoxNumeric Event Structures: A661_EVT_VALUE_CONFIRMED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_VALUE_CONFIRMED
UnusedPad	N/A	16	0
Value	float	32	

Available SetParameter identifiers and associated data structure are defined in Table 3.3.10-6.

Table 3.3.10-6 - 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)	Value
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE
Value	float	32	A661_VALUE	A661_ParameterStructure_4Bytes	
StartCursorPos	ushort	16	A661_CURSOR_POS	A661_ParameterStructure_2Bytes	
StyleSet	ushort	16	A661_STYLE_SET	A661_ParameterStructure_2Bytes	

3.3.11 EditBoxText

Categories:
Graphical representation
Interactive
Text string

Description:
A 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.

3.3.11 EditBoxText (cont'd)

COMMENTARY

The data/formatting displayed for current-item-selected during crew editing and UA validation is part of CDS internal behavior, and is beyond the scope of this document. For instance, the CDS could implement a graphical representation indicating that one text has been edited but not yet validated by the UA. In this case the UA could show that it has validated the entry through a SetParameter command on either the LabelString or EditBoxState parameter, according to defined widget behavior. Similarly, the UA should display an ERROR mode through the use of the StyleSet parameter.

c-1

When the EditBoxText 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.

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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	TRUE CDS will report each update from the crew member while in edit mode (A661_EVT_VALUE_CHANGE) CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_VALUE_CHANGE_ABORTED) FALSE CDS will report the value change after crew member validation (A661_EVT_VALUE_CONFIRMED) Note: In all cases, the CDS will report the entry in edit mode

c-1

3.3.11 EditBoxText (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_CENTER A661_LEFT 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.

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

c-1

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

Available SetParameter identifiers and associated data structure are defined in Table 3.3.11-6.

Table 3.3.11-6 - EditBoxText Runtime Modifiable Parameters

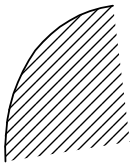
Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)	Value
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte	A661_FALSE A661_TRUE
LabelString	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.

UNFILLED
ARC



FILLED
ARC

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.

3.3.12 GpArcEllipse (cont'd)

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_XY
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
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

3.0 WIDGET LIBRARY

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.

3.3.13 GpArcCircle (cont'd)

GpArcCircle Creation Structure is defined inTable 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	8	0

The GpArcCircle widget does not send any event.

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

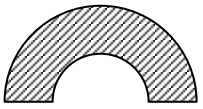
Table 3.3.13-3 - GpArcCircle Runtime Modifiable Parameters

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_XY
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.



Restriction: none

3.3.14 GpCrown (cont'd)

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_XY
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_XY
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_XY
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.

3.3.16 GpLinePolar (cont'd)

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_XY
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

3.0 WIDGET LIBRARY

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.

3.3.17 GpRectangle(cont'd)

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_XY
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
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

3.3.18 GpTriangle (cont'd)

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_XY
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_XY
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_XY
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

3.3.19 Picture (cont'd)

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.

c-1

Restriction: none

3.0 WIDGET LIBRARY

Label Parmaters 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	Font of the displayed string
ColorIndex	DR	Applicable color index if color definition inside StyleSet is FREE_COLOR
Alignment	D	Justification of the label text within the label area BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight

c-1

3.3.20 Label (cont'd)

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.

c-1

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_XY
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

c-1

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 embedded escape sequences, refer to Section 3.2.5.5, Escape Sequences Definition.

Restriction: N/A

3.3.21 LabelComplex (cont'd)

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" For Escape sequences defining the Default style for the text, refer to Section 3.2.5.4, Default Graphic Properties.
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

c-1

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.

c-1

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

c-1 | 3.3.22 MapHorz_ItemList

Categories:
Map management
Graphical Representation
Interactive
Text string

Description:

c-1 | 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.

c-1 | A MapHorz_ItemList must be in a MapHorz_Source container.

c-1 | 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).

c-1 | 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.

c-1 | MapHorz_ItemList Parameters are defined inTable 3.3.22-1.

Table 3.3.22-1 - MapHorz_ItemList Parameters

Parameters	Change	Description
Commonly used parameters		
c-1 | 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		
c-1 | MaxNumberOfItem	D	Maximum number of items that the UA can address under the MapHorz_ItemList.
BufferOfItems	R	Buffer of the Map Items

MapHorz_ItemList Creation Structure is defined in Table 3.3.22-2a. c-1

Table 3.3.22-2a - 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
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfItem	ushort	16	
UnusedPad	N/A	16	0

c-1

MapHorz_ItemList Event Structures: A661_EVT_SELECTION are defined in Table 3.3.22-2b.

Table 3.3.22-2b - MapHorz_ItemList Event Structures: A661_EVT_SELECTION c-1

EventStructure	Size (bits)	Value/Description
EventIdent	16	A661_EVT_SELECTION
Item Index	16	Index of the item which has been selected

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

Table 3.3.22-3 - MapHorz_ItemList Runtime Modifiable Parameters c-1

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_MAPHORZ_ITEM	A661_ParameterStructure_BufferOfItems Refer to “MapHorz_ItemList A661_ParameterStructure Specifics”, Section 3.3.22.2, below.

c-1

3.3.22.1 MapHorz_ItemList Standard Items Description

This section describes the MapHorz item structures.

Table 3.3.22.1-1- 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 a 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). This Item holds a 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 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 than SYMBOL_GENERIC, except orientation and Length parameters are added.

3.3.22.2 MapHorz_ItemList A661_ParameterStructure Specifics c-1

This section describes the A661_ParameterStructure_BufferOfItems.

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
UnusedPad	N/A	16	0
ItemStyleSet	ushort	16	

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
UnusedPad	N/A	8	0
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapHorz_Source MapDataFormat)
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapSource coordinate system (fixed real LSB depends on MapHorz_Source)

c-1

c-1

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
c-1	X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)
c-1	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)	c-1
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)	c-1

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	
Radius	fr(32768)	32	(in nautical miles)
CourseChange	fr(180)	32	

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
UnusedPad	N/A	24	0
SymbolType	uchar	8	EXAMPLES: SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
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)

c-1

c-1

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. approximate LSB = 0.000030517578125 nm

c-1

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	
UnusedPad	N/A	24	0	
SymbolType	uchar	8	EXAMPLES: SYMBOL_HOLD_LEFT SYMBOL_HOLD_RIGHT SYMBOL_PROCEDURE_TURN_LEFT SYMBOL_PROCEDURE_TURN_RIGHT SYMBOL_LONG_RANGE_AIRPORT_WITH_RUNWAY	c-1
X / Lat / Range	Scaled Integer	32	First coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)	c-1
Y / Lng / Bearing	Scaled Integer	32	Second coordinate of symbol, MapHorz_Source coordinate system (fixed real LSB depends on MapDataFormat)	c-1
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)	c-1
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	
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)

c-1

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

3.3.22.2.2 A661_ParameterStructure_BufferOfItems

A661_ParameterStructure_BufferOfItems 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}+	

3.3.23 MapLegacy

Categories:
Map management
Graphical Representation

Description:
c-1 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		
c-1 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	
c-1 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 one 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. c-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	Indicates if the UA wants to receive the cursor position upon click, expressed in its coordinate system.

3.3.24 MapHorz_Source (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
MapDataFormat	uchar	8	A661_MDF_BRG_DIST_ACHDG A661_MDF_LAT_LONG A661_MDF_LEGACY
EventFlag	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	16	0

c-1 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	Units of Measure	LSB
A661_MDF_BRG_DIST_ACHDG	No	aircraft body longitudinal axis	aircraft lat/lng defined in MapHorz	X nautical miles Y degrees	X: fr(32768) Y: fr(180)
A661_MDF_DIST_DIST	No	Various	aircraft lat/lng defined in MapHorz	X and Y: nautical miles	X: fr(32768) Y: fr(32768)
A661_MDF_LAT_LONG	Yes	True North	lat/lng	X and Y degrees	X: fr(180) Y: fr(180)
A661_MDF_LEGACY	various	various	various	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

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

Table 3.3.24-4 - 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

c-1

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

3.0 WIDGET LIBRARY

3.3.25 MapHorz (cont'd)

c-1 MapHorz Parameters are defined in Table 3.3.25-1.

Table 3.3.25-1 - MapHorz Parameters

Parameters	Change	Description
c-1 *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
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	
c-1 *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	
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		
c-1 Orientation	R	Angle of the True North relative to the up-direction of display at the PRP. (see Reference coordinate system, above). 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
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: LAMBERT POLAR

3.3.25 MapHorz (cont'd)

MapHorz Creation is defined in Table 3.3.25-2. c-1

Table 3.3.25-2 - 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
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

c-1

No Event is associated with the MapHorz widget.

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_XY
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_XY
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	64	A661_AC_LAT_LONG	A661_ParameterStructure_XY
AircraftOrientation	fr(180)	32	A661_AC_ORIENTATION	A661_ParameterStructure_4Bytes

c-1 c-1 c-1

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

3.0 WIDGET LIBRARY

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
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

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	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

3.0 WIDGET LIBRARY

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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

c-1

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	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.

c-1

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

3.3.28 PicturePushButton (cont'd)

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

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on a widget having the following FocusIndex value.
Specific parameters		
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

c-1

3.3.29 PictureToggleButton (cont'd)

PictureToggleButton Creation Structureis 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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	ushort	16	
MaxStringLength	ushort	16	
AlternateFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
PictureReference	ushort	16	
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.

c-1

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	Size (bits)	Value/Description
EventIdent	16	A661_EVT_STATE_CHANGE
UnusedPad	8	0
ToggleState	8	A661_UNSELECTED A661_SELECTED

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

3.0 WIDGET LIBRARY

3.3.30 PopUpPanel

Categories:
Container
Graphical representation
Interactive

Description:
PopUpPanel widgets should be displayed on the top of other layer, but it is affected by clipping area of its parents.
PopUpPanel widget invisibility should be managed by the CDS through logic defined by the airframe manufacturer/system integrator.
PopUpPanel widgets 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 widgets has clipping capability.

Restriction:
PopUpPanel widgets can not be nested.

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 Widget is not visible at creation time.
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

c-1

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	

c-1

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_CLOSED

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_POPUP_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

3.0 WIDGET LIBRARY

3.3.31 PopUpMenu

Categories:
Graphical representation
Interactive
Text string

Description:
The PopUpMenu widget should be displayed on the top of other layer, but it is affected by clipping area of its parents. PopUpMenu is not a Container. PopUpMenu invisibility 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 PositionFlag 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 Widget 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
MaxStringLength	D	Maximum length of the text of any one entry.
StringArray	DR	String attached to one entry NULL string will be interpreted as “separator.” The NULL string at the end of the array will be not interpreted.
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

c-1

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 PosY	long long	32 32	Set to 0 when UAPositionFlag is FALSE
SizeX	ulong	32	
SizeY	ulong	32	
MaxStringLength	ushort	16	Maximum length of the text of one entry
StyleSet	ushort	16	
PopUpIdentArray[]	{ushort}+	16 * NumberOf Entries	
EnableArray[]	{uchar}+	8 * NumberOf Entries	
StringArray[]	{string}+	8 * string lengths + Pad	Each string terminating NULL is used as string separator.

c-1

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

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 [Entry]	N/A	{32}+	A661_ENABLE_ARRAY	A661_ParameterStructure_EnableArray 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_EnableArray is defined in Table 3.3.31.1-1.

Table 3.3.31.1-1 - A661_ParameterStructure_EnableArray

A661_ParameterStructure_EnableArray	Size (bits)	Description
Parameter_ident	16	A661_ENABLE_ARRAY
EntryIndex	8	
Enable	8	A661_FALSE A661_TRUE

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 Entry Updated	16	
{ EntryPopUp_Structure }+	{32}+	

EntryPopUp_Structure is defined in Table 3.3.31.1-3.

Table 3.3.31.1-3 - EntryPopUp_Structure

EntryPopUp_Structure	Size (bits)	Description
UnusedPad	8	0
EntryIndex	8	
Enable	8	A661_FALSE A661_TRUE
StringLength	8	
String	{32}+	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned.

3.0 WIDGET LIBRARY

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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 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 CDS_DEPENDENT:

3.0 WIDGET LIBRARY

Parameters	Change	Description
		Position defined by CDS using CCD Click location
NumberOfEntries	D	Number of entries in the PopUpMenu
MaxStringLengthPopUp	D	Maximum string length for the entries on the popup
StringArray	DR	String attached to one entry NULL string will be interpreted as “separator.” The NULL string at the end of the array will be not interpreted.
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: ENABLE DISABLE

c-1

3.3.32 PopUpMenuButton (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	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 A661_CDS_DEPENDENT
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UnusedPad	N/A	8	0
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
LabelString	string	8 * string length + Pad	Followed by zero, one, two or three NULL character(s) to be 32 bits aligned
PopUpIdentArray[]	{ushort}+	16 * NumberOf Entries	
EnableArray[]	{uchar}+	8 * NumberOf Entries	
StringArray[]	{string}+	8 * string length + Pad	Each string terminating NULL is used as string separator.

c-1

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
UnusedPad	N/A	8	0
SelectedEntry	uchar	8	0 when the pop up is closed without any selection 'n' in [1; NumberOfEntry] else.

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_1Byte
String	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 [Entry]	N/A	{32}+	A661_ENABLE_ARRAY	A661_ParameterStructure_EnableArray 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

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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

c-1

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	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

c-1

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	Size (bits)	Value/Description
EventIdent	16	A661_EVT_SELECTION
UnusedPad	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

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. Never the less, 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.

c-1

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

ToggleButton
PictureToggleButton
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 (one and only one is selected) at all times, including when the user changes the state of the children. The CDS prevents UAs from deselecting the selected child of a RadioBox (this is not a normal operation). The change of child state generates two events, one for select and one for deselect.

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
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
Enable	DR	Ability of the widget to be activated
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

3.3.35 RotationContainer (cont'd)

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	
Enable	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
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)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
CenterX CenterY	long x 2	32 x 2	A661_CENTER_XY	A661_ParameterStructure_XY
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

3.0 WIDGET LIBRARY

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 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.
c-1 Two parameters are available to allow the UA to choose from a variety of positions according to the airframe manufacturer/system integrator style guide.

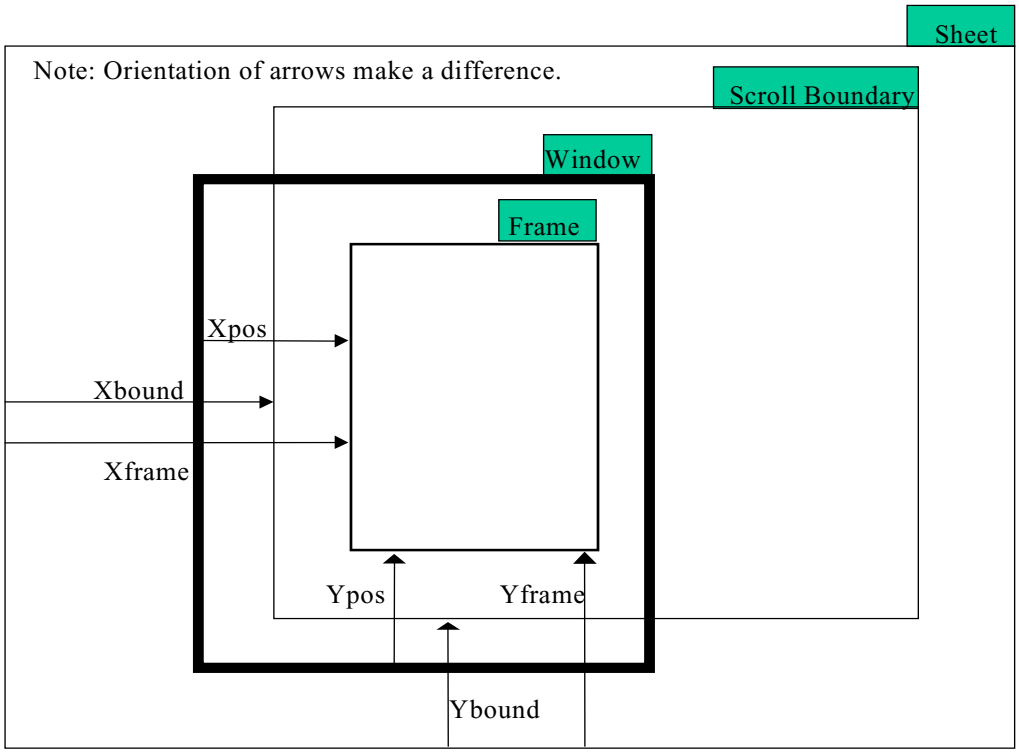


Figure 3.3.36 Frame Standard Coordinate System

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

3.3.36 ScrollPanel (cont'd)

ScrollPanel Parameters are defined in Table 33.3.36-1.

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
StyleSet	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

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
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

c-1

3.3.36 ScrollPanel (cont'd)

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
BoundX	long	32	A661_BOUND_X	A661_ParameterStructure_4Bytes
BoundY	long	32	A661_BOUND_Y	A661_ParameterStructure_4Bytes
SizeXbound	ulong	32	A661_BOUND_SIZE_X	A661_ParameterStructure_4Bytes
SizeYbound	ulong	32	A661_BOUND_SIZE_Y	A661_ParameterStructure_4Bytes

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. The size of the selection area is determined by graphical parameters (size of widget, number of entries), and not by the text. The UA should verify that the text fits in the selection area.

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

3.3.37 ScrollList (cont'd)

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
	FocusIndex	D	Order of the widget for focus circulation
	AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
	Specific parameters		
	NumberOfEntries	DR	Number of accessible entries
	MaxNumberOfEntries	D	Max number of entries that can be managed by objects
	FirstVisibleEntry	DR	Index of the entry appearing on top
c-1	ShiftFirstVisibleEntry	R	Index shift to be applied to the current first visible entry. That is, when a UA wants to move the entries down and insert new ones above, this parameter allows the UA to offset indices while new data is loading, to accommodate for the crossover.
c-1	FirstAccessibleEntry	DR	Entries with lower indices do not appear. Entries with higher indices than FirstAccessibleEntry plus NumberOfEntry do not appear.
	FlagReportVisibleEntry	D	If True, CDS will report change on first visible entry following crew member actions.
	SelectedEntry	DR	Currently selected Entry index
	DefaultStyleText	D	NULL character: Escape sequence not used, entries in the ScrollList are simple labels. "ToutLine⊗TBackColor⊗TForeColor⊗TFont" Escape sequences defining the Default style for the text, refer to Section 3.2.5.4, Default Graphic Properties.
	MaxStringLength	D	Maximum string length able to be received by the object
c-1	Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
	LabelStringArray[MaxNumberOfEntries]	DR	Label array of the ScrollList
	Vertical Scroll	D	ABSENT LEFT RIGHT TOP BOTTOM

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	
Visible	uchar	8	A661_FALSE A661_TRUE	
PosX	long	32		
PosY	long	32		
SizeX	ulong	32		
SizeY	ulong	32		
StyleSet	ushort	16		
FocusIndex	ushort	16		
MaxNumberOfEntries	ushort	16		
NumberOfEntries	ushort	16		
SelectedEntry	ushort	16		
Vertical Scroll	uchar	8	A661_LEFT A661_RIGHT A661_ABSENT	c-1
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER	
FirstAccessibleEntry	ushort	16		
FirstVisibleEntry	ushort	16		
MaxStringLength	ushort	16		
FlagReportVisibleEntry	uchar	8	A661_FALSE A661_TRUE	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE	
DefaultStyleText	uchar	96		
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.	

3.3.37 ScrollList (cont'd)

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	Size (bits)	Value/Description
EventIdent	16	A661_EVT_SEL_ENTRY_CHANGE
SelectedEntry	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	Size (bits)	Value/Description
EventIdent	16	A661_EVT_FIRST_VIS_ENTRY_CHANGE
FirstVisibleEntry	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
FirstAccessibleEntry	ushort	16	A661_FIRST_ACCESS_ENTRY	A661_ParameterStructure_2Bytes
FirstVisibleEntry	ushort	16	A661_FIRST_VISIBLE_ENTRY	A661_ParameterStructure_2Bytes
ShiftFirstVisibleEntry	ushort	16	A661_SHIFT_FIRST_VISIBLE_ ENTRY	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	16	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
LabelStringArray	N/A	{32}+	A661_STRING_ARRAY	A661_ParameterStructure_StringArray

3.0 WIDGET LIBRARY

3.3.38 Symbol

Categories: c-1
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

3.3.38 Symbol (cont'd)

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	
c-1 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_XY
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
c-1 SymbolReference	ushort	16	A661_SYMBOL_REFERENCE	A661_ParameterStructure_1Byte

3.0 WIDGET LIBRARY

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 TabbedPanel 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 FocusIndex 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	
FocusIndex	D	Order of the widget for focus circulation	
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.	
Specific parameters			
LabelString	DR	Label of the tab	
Alignment	D	Alignment of the text within the label area of the widget LEFT RIGHT CENTER	c-1
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).	c-1
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	

3.0 WIDGET LIBRARY

3.3.39 TabbedPanel (cont'd)

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 children.

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
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
FocusIndex	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.

c-1

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

3.0 WIDGET LIBRARY

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 Panels inside the TabbedPanel widgets occupy the same display space, and only one may be displayed at a time. This TabbedPanel is the one referenced by the "Active TabbedPanel ID".

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 buttonSize. 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 inside the TabbedPanel
ActiveTabbedPanelID	DR	Identifier of the active TabbedPanel

c-1

3.3.40 TabbedPanelGroup (cont'd)

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
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

c-1

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 new selected TabbedPanel

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

3.0 WIDGET LIBRARY

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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value.
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 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

c-1

3.3.41 ToggleButton (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	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 * string1 length	
AlternateLabelString	string	8 * string2 length + Pad	Followed by zero, one, two or three extra NULL for alignment of 32 bits.

c-1

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
UnusedPad	N/A	8	0
InnerState	uchar	8	A661_UNSELECTED A661_SELECTED

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

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
Enable	DR	Ability of the widget to be activated
Specific parameters		
TranslationX	DR	X Translation of the child widgets
TranslationY	DR	Y Translation of the child widgets

3.3.42 TranslationContainer (cont'd)

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	
Enable	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
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)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte
TranslationX TranslationY	long x 2	32 x 2	A661_TRANSLATION_XY	A661_ParameterStructure_XY
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

c-1

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
- Is there is sufficient memory allocated in the CDS for an oversized array

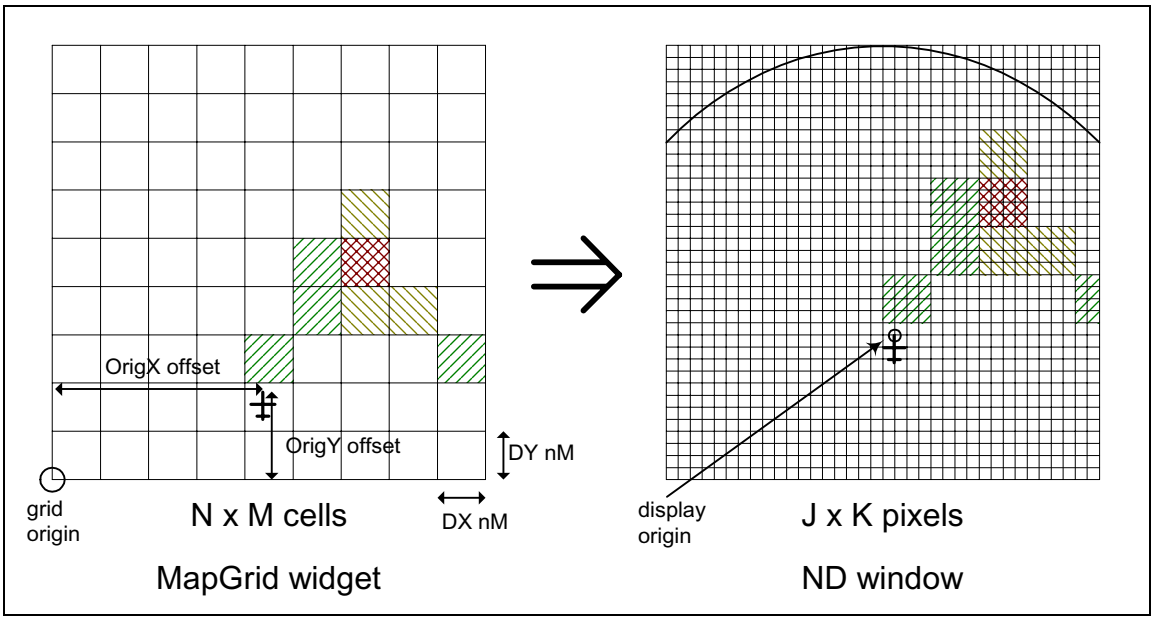


Figure 3.4.1-1 Example MapGrid rendering in ND window

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 container.

c-1

3.0 WIDGET LIBRARY

3.4.1 MapGrid (cont'd)

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 MapHorz_Source: A661_MDF_LAT_LONG: IncrementX is in degrees of longitude A661_MDF_BRG_DIST_ACHDG: IncrementX is in nautical miles For MapVert sources: IncrementX is in nautical miles
IncrementY	DR	Size of each individual cell in the Y axis, in the real-world units defined by the containing MapHorz_Source: A661_MDF_LAT_LONG: IncrementY is in degrees of latitude A661_MDF_BRG_DIST_ACHDG: IncrementY is in nautical miles For MapVert sources: IncrementY is in feet
BufferOfFillStyles	R	Buffer of Fill Style Indices. Buffer can be updated row-at-a-time or column-at-a-time.

3.0 WIDGET LIBRARY

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.

c-1

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	32 x 2	A661_MAPGRID_OFFSET	A661_ParameterStructure_ XY
IncrementX, IncrementY	long	32 x 2	A661_MAPGRID_CELLSIZE	A661_ParameterStructure_ XY
BufferOfFillStyles	float	32	A661_BUFFER_OF_FILL_STYLES	A661_ParameterStructure_BufferOfFillStyles Refer to “MapGrid Parameter Structure Specifics” section below.

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), 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

A661_ParameterStructure_BufferOfFillStyles are defined in Table 3.4.1.1-1.

Table 3.4.1.1-1 - A661_ParameterStructure_BufferOfFillStyles

Field	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
Unused Pad	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.

c-1

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 these 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

c-1

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 re-size or rotate video signal received.

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

3.4.2 ExternalSource (cont'd)

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.

c-1

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
Unused	uchar	8	
X Pos	long	32	
Y Pos	long	32	
SizeX	long	32	
SizeY	long	32	
SourceReference	ushort	16	
Unused	ushort	16	
SourceX	ulong	32	
SourceY	ulong	32	
SourceDX	ulong	32	
SourceDY	ulong	32	
StyleSet	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

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 passed to the CDS can be converted in screen distance on both axes. c-1
- 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.

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 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

3.4.3 MapVert (cont'd)

MapVert Creation Structure is defined in Table 3.4.3-2.

Table 3.4.3-2 - 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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	long	32	
SizeY	long	32	
RangeX	fr(32768)	32	
RangeY	$fr(2^{31}) = 1$	32	
RefPosX	long	32	
RefPosY	long	32	
RefGeoPosX	fr(32768)	32	
RefGeoPosY	$fr(2^{31}) = 1$	32	

c-1 No Event is associated with the MapVert widget.

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	ulong	32	A661_RANGE_X	A661_ParameterStructure_4Byte
RangeY	ulong	32	A661_RANGE_Y	A661_ParameterStructure_4Byte
RefPosX	long	32	A661_PRP_SCREEN_X	A661_ParameterStructure_4Byte
RefPosY	long	32	A661_PRP_SCREEN_Y	A661_ParameterStructure_4Byte
RefGeoPosX	long	32	A661_PRP_X	A661_ParameterStructure_4Byte
RefGeoPosY	long	32	A661_PRP_Y	A661_ParameterStructure_4Byte

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.

c-1

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	A661_Relative: X position of Items expressed relative to Reference point. A661_Absolute: X position of Items expressed in absolute value.
MapDataFormatY	D	A661_Relative: Y position of Items expressed relative to Reference point. A661_Absolute: Y position of Items expressed in absolute value.
EventFlag	DR	Indicates if the UA wants to receive the cursor position upon click, expressed in its coordinate system.

3.4.4 MapVert_Source (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
MapDataFormatX	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE
MapDataFormatY	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE
EventFlag	uchar	8	A661_FALSE A661_TRUE
Unused	uchar	8	

c-1

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

Table 3.4.4-2b - MapVert_MapData Format Values

Parameter	Type	Size (bits)	Value	Origin	Units of Measure	LSB
MapDataFormatX	long	32	A661_MDF_ABSOLUTE	RefGeoPosX	nm	fr(32768)
MapDataFormatX	long	32	A661_MDF_RELATIVE	RefPosX	nm	fr(32768)
MapDataFormatY	long	32	A661_MDF_ABSOLUTE	RefGeoPosY	feet	1
MapDataFormatY	long	32	A661_MDF_RELATIVE	RefPosY	feet	1

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	Scaled Integer	32	expressed in map source coordinate system
Y	Scaled Integer	32	expressed in map source coordinate system

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

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: c-1
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

3.4.5 MapVert_ItemList (cont'd)

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
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfItem	ushort	16	
UnusedPad	N/A	16	0

c-1 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	Size (bits)	Value/Description
EventIdent	16	A661_EVT_SELECTION
Item Index	16	Index of the item that has been selected

Available SetParameter identifiers and associated data structure are defined in Table 3.4.5-3.

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_ITEM	A661_ParameterStructure_BufferOfItems Refer to MapVert_ItemList A661_ParameterStructure Specifics Section 3.4.5.2

3.4.5.1 MapVert_ItemList Standard Items Description

This section describes all the Item structures.

Table 3.4.5.1-1 - 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.
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.
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_RUNWAY	Same than SYMBOL_GENERIC except Length parameter is added

c-1

3.4.5.2 MapVert_ItemList A661_ParameterStructure Specifics

This section describes the A661_ParameterStructure_BufferOfItems.

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
UnusedPad	N/A	16	0
ItemStyleSet	ushort	16	

c-1

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
UnusedPad	N/A	8	0
X	Scaled Integer	32	First coordinate of symbol, (fixed real LSB depends on MapVert_Source MapDataFormat)
Y	Scaled Integer	32	Second coordinate of symbol, (fixed real LSB depends on MapVert_Source)

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 3.4.5.21..4.

Table 3.4.5.2.1.4 - Line_Start

c-1

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

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	Scaled Integer	32	First coordinate of symbol, (fixed real LSB depends on MapVert_Source MapDataFormat)
Y	Scaled Integer	32	Second coordinate of symbol, (fixed real LSB depends on MapVert_Source)

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
UnusedPad	N/A	24	0
SymbolType	uchar	8	SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
X	Scaled Integer	32	First coordinate of symbol, (fixed real LSB depends on MapVert_Source MapDataFormat)
Y	Scaled Integer	32	Second coordinate of symbol, (fixed real LSB depends on MapVert_Source)

c-1

3.4.5.2.1.8 Symbol_Runway

Symbol_Runway is defined in Table 3.4.5.1.8.

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	Scaled Integer	32	First coordinate of symbol, (fixed real LSB depends on MapVert_Source MapDataFormat)
Y	Scaled Integer	32	Second coordinate of symbol, (fixed real LSB depends on MapVert_Source)
Length	fr(32768)	32	Length of runway

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	Scaled Integer	32	First coordinate of symbol (LSB and units defined by MapVert_Source)
Y / Lng / Angle / Alt	Scaled Integer	32	Second coordinate of symbol (LSB and units defined by MapVert_Source)

c-1

3.4.5.2.2 A661_ParameterStructure_BufferOfItems

A66l_ParameterStructure_BufferOfItems 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_ITEM
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}+	

3.4.6 EditBoxMultiLine

Categories:
Graphical representation
Interactive
ASCII Text

3.4.6 EditBoxMultiLine (cont'd)

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.

COMMENTARY

The information displayed, as the current one selected, after a crew member selection and validation is part of CDS internal behavior, and is beyond the scope of this document. For instance, The CDS can implement a graphical representation indicating that one text has been edited but this value has not been validated yet by the UA. In this case the UA could validate the selection through a setParameter command on the LabelString parameter. In case of invalid string edit by a crew member, the UA should display an ERROR mode through the use of the SyleSet parameter.

EditBoxMultiLine Paramters 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
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocus Motion	D	Automatic motion of the focus on widget having the following FocusIndex value.
Specific parameters		
MaxStringLength	D	Maximum length of the entire text included the first NULL character ended the string (in bytes), MaxStringLength > 1.
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

Note: D is Design time. R is Run time

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
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
FocusIndex	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_FALSE A661_TRUE
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.

c-1

The specific event sent by the EditBoxMultiLine to the owner application is defined in Table 3.4.6-3.

Table 3.4.6-3 - 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.

Available SET_PARAMETER identifiers and associated data structure are:

3.4.6 EditBoxMultiLine (cont'd)

EditBoxMultiLine Runtime Modifiable Parameters is defined in Table 3.4.6-4.

Table 3.4.6-4 - 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

3.4.7 ComboBoxEdit

Categories:
Graphical representation
Interactive
ASCII Text

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. For example, click on the arrow button associated to the Selected Entry.

c-1 Moreover, ComboBoxEdit allows the crew to enter the static part a new item. 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 Popup part of the ComboBoxEdit

OpeningMode of the ComboBoxEdit is used to determine how the ComboBox opens.

ComboBoxEdit has to be displayed on top of its layer and is affected by clipping area of its layer.

COMMENTARY

The information displayed, like the current selected entry, upon a crew member interaction, is part of CDS internal behavior, and is beyond the scope of this document. For instance, The CDS can implement a graphical representation indicating that the item has been selected but the item has not yet been validated by the UA. In this case the UA could validate the selection through a setParameter command on the SelectedEntry parameter.

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)
FocusIndex	D	Order of the widget for focus circulation
AutomaticFocusMotion	D	Automatic motion of the focus on widget having the following FocusIndex value
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	TRUE CDS will report each update from the crew member (A661_EVT_STRING_CHANGE) CDS will report the string change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report an event if the crewmember aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) FALSE CDS will report the string change only after crew member validation (A661_EVT_STRING_CONFIRMED)
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
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.

c-1

3.4.7 ComboBoxEdit (cont'd)

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
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	
FocusIndex	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
UnusedPad	N/A	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_FALSE A661_TRUE
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_SELECTED_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.

c-1

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.

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.4.7-7 - ComboBoxEdit Runtime Modifiable Parameters

Name of the parameter to set	Type	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Enable	uchar	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	A661_VISIBLE	A661_ParameterStructure_1Byte
NumberOfEntries	ushort	A661_NUMBER_OF_ENTRIES	A661_ParameterStructure_2Bytes
SelectedEntry	ushort	A661_SELECTED_ENTRY	A661_ParameterStructure_2Bytes
StyleSet	ushort	A661_STYLE_SET	A661_ParameterStructure_2Bytes
StartCursorPos	ushort	A661_CURSOR_POS	A661_ParameterStructure_2Bytes
EntryList [NumberOfEntries]	N/A	A661_STRING_ARRAY	A661_ParameterStructure_StringArray

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

Illustration of possible behavior for 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. This behavior is different from the behavior of a stand-alone PopUpMenu. Right / Left arrow keys move the focus from one button to another button of the MenuBar.

c-1 The Buttons contained in the MenuBar will be individually defined with the MenuBar as parent widget. The positions of buttons inside the MenuBar is defined by their own parameters PosX and PosY, according to the following rules:

For Horizontal menu bar, all Button 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		True: MenuBar is horizontal False: MenuBar is Vertical
ButtonPos		If Horizontal =True: Value of the buttons parameter PosY If Horizontal =False: Value of the buttons parameter PosX
ButtonSize		If Horizontal =True: Value of the buttons parameter SizeY If Horizontal =False: Value of the buttons parameter SizeX

MenuBar Creation Structure is defined in Tablwe 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
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	

c-1

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	ParameterIdent used in the ParameterStructure	Type of Structure Used (Refer to 4.5.4.5)
Visible	uchar	A661_VISIBLE	A661_ParameterStructure_1Byte
Enable	uchar	A661_ENABLE	A661_ParameterStructure_1Byte

4.0 COMMUNICATION PROTOCOL

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 provided 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

One DF contains User Application Layer Definitions (UALDs) from one UAs. The UALD describes data shared between one 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 inside one 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 insured by the use of “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 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.

4.0 COMMUNICATION PROTOCOL

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 one 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, allowing both data loading and downloading, is a standard non-executable binary format.

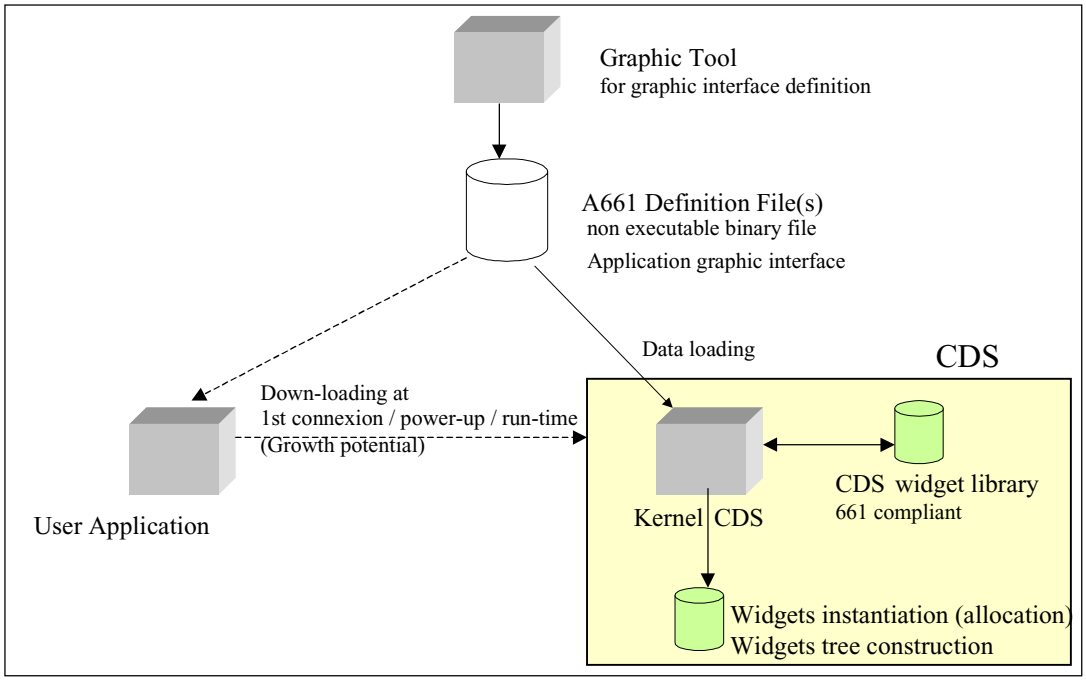


Figure 4.1 Definition File Integration Process

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

4.0 COMMUNICATION PROTOCOL

4.3.1 General Principle (cont'd)

values. Nevertheless, if synchronous refresh of information is needed for one UA, this one UA could send this kind of information on a periodic basis.

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 one 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. Tefer 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 hypothesis applies to:

The correct order reception of messages, Issue 2

The loss of messages, Issue 1

4.3.4 Layer Data Management

Refer to Section 2.3.2.4, Layer Activity/Visibility Management.

4.0 COMMUNICATION PROTOCOL

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.

4.0 COMMUNICATION PROTOCOL

4.4.2 Error Notification

This Specification defines the principle of error notification to provide the tools to manage error*s* in a command. Nevertheless it is outside the scope of this document to define the recovery action on an error notification. Aircraft OEM should specify only error recording in built-in test equipment (BITE), or some recovery mechanism to implement specific errors.

The error notifications with commands in this specification are described in the Table 4.4.2. In addition to the error notification defined in response to an incorrect command, the CDS should send an error notification to notify an overload.

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.

4.0 COMMUNICATION PROTOCOL

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 the 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

Request from the UA to the CDS, described in Table 4.4.3.1, may or may not be accepted by the CDS. This request 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.

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.

4.0 COMMUNICATION PROTOCOL

4.5 ARINC 661 Command Structure

4.5.1 Notation

Notations used in ARINC 661 Command Structures:

<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

4.5.2 Block Structure

UA and CDS exchange information through a block of commands. One 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 Definition File (DF) Structure

A DF may hold several UALD from one or more UA. One UA may define several DF. The loadable entity is the DF. Table 4.5.3.1-1 describes the structure of one DF.

Table 4.5.3.1-1 Definition File Structure

A661_Definition_File	Description
OEM File Header	Header of the file
{ A661_Block_Structure_DT }+	ARINC 661 core of the file
OEM File Footer	Footer of the file

The DF should have a header part, which is OEM specific. Version number should to support version compatibility checks, which may be required by Original Equipment Manufacturer (OEM) aircraft. The DF should have a footer part, which is also OEM specific.

4.0 COMMUNICATION PROTOCOL

Block Structure Exchanged Between UA and CDS at Definition Time is defined in Table 4.5.3.1-2.

Table 4.5.3.1-2 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 one layer. This value will be attached by 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

4.5.3.2 Definition Time Block Commands

One or more A661_Definition_Command can be included in a block. Table 4.5.3.2 lists the commands defined in the protocol.

Table 4.5.3.2 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.

4.5.4 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.
UnusedPad	N/A	16	0
WidgetIdent	ushort	16	Identifier of the widget
{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.

4.5.4.2 Command Structure – Run-Time Comamnds (cont'd)

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.

4.0 COMMUNICATION PROTOCOL

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

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

Table4.5.4.5.1-1 ParameterStructure_1Byte

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
UnusedPad	N/A	8	0
ParameterValueBuffer	uchar	8	Values associated with the parameter type

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 Parameter Structure

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_Cell Structure

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_XY

Table 4.5.4.5.7-1 ParameterStructure_XY

A661_ParameterStructure	Type	Size (bits)	Description
Parameter_ident	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
ParameterValue1	long	32	PosX
ParameterValue2	long	32	PosY

4.5.4.5.7 A661_ParameterStructure_BufferOfItems

Refer to MapHorz_ItemList widget description in Section 3.3.22.2. c-1

4.5.4.5.8 A661_ParameterStructure_Buffer

Refer to BufferFormat widget description in Section 3.3.4.1.

4.5.4.5.9 A661_ParameterStructure_EntryPopUpArray

Refer to PopUpMenu widget description in Section 3.3.31.1.

4.5.4.5.10 A661_ ParameterStructure_EntryPopUpArray

Refer to PopUpMenu widget description in Section 3.3.31.1.

4.0 COMMUNICATION PROTOCOL

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

Table4.6-2 Block Codes (8 bits)

ARINC 661 Block Codes (8 bits)	
A661_BEGIN_LAYER_BLOCK	0xA0
A661_BEGIN_BLOCK	0xB0
A661_END_LAYER_BLOCK	0xC0
A661_END_BLOCK	0xD0

Table 4.6-3 Commands

ARINC 661 Commands	
A661_CMD_CREATE	0xCA01
A661_CMD_SET_PARAMETER	0xCA02
A661_CMD_UA_REQUEST	0xCA03

Table 4.6-4 Notifications

ARINC 661 Notifications	
A661_NOTIFY_EXCEPTION	0xCC03
A661_NOTIFY_LAYER_EVENT	0xCC02
A661_NOTIFY_WIDGET_EVENT	0xCC01

4.0 COMMUNICATION PROTOCOL

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_NOTE_REINIT_LAYER_DATA	0xDC01
A661_NOTE_LAYER_IS_ACTIVE	0xDC02
A661_NOTE_LAYER_IS_INACTIVE	0xDC03

Table 4.6-6 Expection Type

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

4.0 COMMUNICATION PROTOCOL

4.6 ARINC 661 Keyword Values (cont'd)

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
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_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
c-1 A661_MAP_HORZ_ITEMLIST	0xA180
A661_MAP_LEGACY	0xA190
c-1 A661_MAP_HORZ_SOURCE	0xA1A0
A661_MAP_HORZ	0xA1B0
A661_MASK_CONTAINER	0xA1C0
A661_PANEL	0xA1F0
A661_PICTURE	0xA200
A661_PICTURE_PUSH_BUTTON	0xA240
A661_PICTURE_TOGGLE_BUTTON	0xA250
A661_POP_UP_MENU	0xA270
A661_POP_UP_MENU_BUTTON	0xA280
A661_POP_UP_PANEL	0xA290
A661_PUSH_BUTTON	0xA2A0
A661_RADIO_BOX	0xA2B0
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
A661_TRANSLATION_CONTAINER	0xA360
A66l_MAP_GRID	0xA178
A661_EXTERNAL SOURCE	0xA188
A661_MAP_VERT	0xA198
c-1 A661_MAP_VERT_SOURCE	0xA1A8
A661_MAP_VERT_ITEMLIST	0xA1B8
A661_EDIT_BOX_MULTI_LINE	0xA1C8
A661_COMBO_BOX_EDIT	0xA1D8
A661_MENU_BAR	0xA1E8

4.0 COMMUNICATION PROTOCOL

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
A661_ALTERN_PICTURE_REFERENCE	0xB080
A661_BLINKING_TYPE	0xB0A8
A661_BOUND_X	0xB0B0
A661_BOUND_Y	0xB0C0
A661_BOUND_SIZE_X	0xB0E0
A661_BOUND_SIZE_Y	0xB0F0
A661_BUFFER_OF_PARAM	0xB110
A661_BUFFER_OF_MAPITEM	0xB120
A661_CENTER_X	0xB130
A661_CENTER_XY	0xB150
A661_CENTER_Y	0xB140
A661_COLOR_INDEX	0xB160
A661_CURSOR_POS	0xB170
A661_ENABLE	0xB180
A661_ENABLE_ARRAY	0xB190
A661_END_ANGLE	0xB1B0
A661_ENTRY_POP_UP_ARRAY	0xB1C0
A661_EVENT_FLAG	0xB1D0
A661_FILL_INDEX	0xB1E0
A661_FIRST_ACCESS_ENTRY	0xB1F0
A661_FIRST_VISIBLE_ENTRY	0xB200
A661_FRAME_X	0xB210
A661_FRAME_Y	0xB220
A661_INNER_RADIUS	0xB240
A661_INNER_STATE_CHECK	0xB244
A661_INNER_STATE_EDIT	0xB248
A661_INNER_STATE_TOGGLE	0xB258
A661_LINE_LENGTH	0xB270
A661_MASK_REFERENCE	0xB280
A661_MASK_ENABLED	0xB290
A661_NUMBER_OF_ENTRIES	0xB2A0
A661_NUMERIC_MASK	0xB2B0
A661_ORIENTATION	0xB2C0
A661_OUTER_RADIUS	0xB2D0
A661_PICTURE_REFERENCE	0xB2F0

4.0 COMMUNICATION PROTOCOL

4.6 ARINC 661 Keyword Values (cont'd)

Table 4.6-8 Parameter Types (cont'd)

ARINC 661 Parameter Types (16 bits) (cont'd)	
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_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_SIZE_X	0xB450
A661_SIZE_Y	0xB460
A661_START_ANGLE	0xB480
A661_STRING	0xB490
A661_STRING_ALTERNATE	0xB498
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_BUFFER_OF_FILL_STYLES	0xB0F8
A661_MAPGRID_CELLSIZE	0xB274
A661_MAPGRID_OFFSET	0xB278

c-1

4.0 COMMUNICATION PROTOCOL

Table 4.6-9 Event Types

ARINC 661 Event Types (16 bits)	
A661_EVT_CURSOR_POS_CHANGE	0xE010
A661_EVT_FIRST_VIS_ENTRY_CHANGE	0xE020
A661_EVT_FRAME_POS_CHANGE	0xE030
A661_EVT_TABBED_PANEL_CHANGE	0xE0B0
A661_EVT_POPUP_CLOSED	0xE040
A661_EVT_SEL_ENTRY_CHANGE	0xE050
A661_EVT_SELECTION	0xE060
A661_EVT_SELECTION_MAP	c-1 0xE068
A661_EVT_STATE_CHANGE	0xE070
A661_EVT_STRING_CHANGE	0xE080
A661_EVT_STRING_CONFIRMED	0xE0A0
A661_EVT_VALUE_CHANGE	0xE0C0
A661_EVT_VALUE_CONFIRMED	0xE0E0

Table 4.6-10 Boolean Constant Values

ARINC 661 Boolean Constant Values	
A661_FALSE	0x00
A661_TRUE	0x01

4.0 COMMUNICATION PROTOCOL

4.6 ARINC 661 Keyword Values (cont'd)

Table 4.6-11 Integer Constant Values

ARINC 661 Integer Constant Values	
On 8 bits	
A661_UNSELECTED	0x00
A661_SELECTED	0x01
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_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
c-1 A661_FILLED_POLY_START	0x2B
A661_SYMBOL_OVAL	0x2C
A661_MDF_BRG_DIST_ACHDG	0x60
A661_MDF_LAT_LONG	0x61
A661_MDF_LEGACY	0x62
A661_MDF_ABSOLUTE	0x65
c-1 A661_MDF_RELATIVE	0x66
A661_MDF_DIST_DIST	0x67
On 16 bits	
A661_STYLE_SET_DEFAULT	0x0000

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 layers descriptions (UALD) expected to be displayed on the CDS according to this UA.
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 + 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.
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).

APPENDIX A
GLOSSARY

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, FM map with interactive way-points, TCAS information, Terrain or weather display, Others. 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 definition of 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.
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.

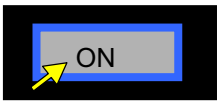
APPENDIX A
GLOSSARY

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
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
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 Overview

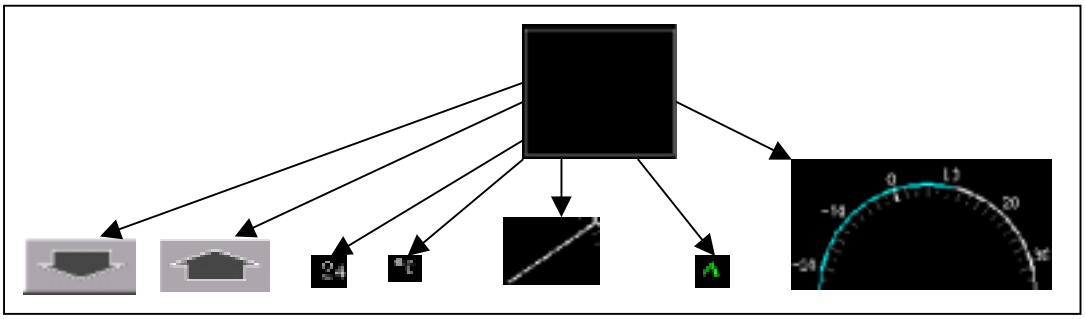
The following example is a simple UA that controls the cabin temperature using two interfaces:

1. The UA is connected to the aircraft environment with:
- one input: a cabin temperature sensor
- one output: an actuator for the heater system and cooler system
2. The UA is connected to the CDS with a ARINC 661 link 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 8 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).

Each widget is a node in a hierarchical structure defined in the DF. The tree for Example C.1is:



APPENDIX C
EXAMPLE OF A DEFINITION FILE

C.2 Example of Application Code at Run time

An example of UA code at run time follows:

```
if ((selectedCabinTemp < maxValue) and (BUTTON_PRESSED.id = IncreaseSelectTemp)) then
        increment(selectedCabinTemp);
end if
if ((selectedCabinTemp > minValue) and (BUTTON_PRESSED.id = IncreaseSelectTemp)) then
        increment(selectedCabinTemp);
end if
Actuator.heaterCommand = PIDcontroller (selectedCabinTemp);
angleValue = selectedCabinTemp * scaleFactor + offset;
setParameter(TemperatureSelectedPointer, RotationAngle, angleValue);
if IsValid(cockpitTemp) = True then
        cabinTemp = Sensor.cabinTemp
        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 a set of data to be processed as coherent information. Hereunder the corresponding byte stream send for this block by the UA on the network, assuming that the bus allows to transport blocks of such size. In other cases, sub-blocking 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
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
4567            # WIDGET ID (DecreaseSelectTemp)

                # Word 8
B180            # PARAMETER ID (A661_ENABLE)
0000            # VALUE (A66_FALSE)

                # Word 9
D0              # A661_END_BLOCK
000000          # UNUSED
```

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

                # Word 6
D0              # A661_END_BLOCK
000000          # UNUSED
```

The UA then acknowledges the event by sending an acknowledgment back to the CDS.

APPENDIX C
EXAMPLE OF A DEFINITION FILE

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. Reminder: unit length of measure is 1/100 of millimeter. Header and Footer appear in *Italic* font style, since they are not defined by the ARINC 661 standard.

```
# START DF
# Input file: cabin_temperature.xml
# Hexadecimal                 #  Comment

                              # Word 1
A661                          # A661_DF_MAGIC_NUMBER
0001                          # A661 VERSION NUMBER

                              # Word 2
6788                          # DF ID
0000                          # UNUSED

                              # Word 3
A0                            # A661_BEGIN_LAYER_BLOCK
42                            # LAYER ID
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
0000                          # UNUSED

                              # 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)
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
1221            # PARENT ID (CabinTempPanel)
00              # UNUSED
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 = 0.0005439]
0001            # StyleSet, value: OEM_STYLESET_FREE_COLOR
9874            # PictureReference, value:SymbolTemperatureIndicatedPointer
0F              # ColorIndex, value: OEM_WHITE
000000          # UNUSED

                # 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              # UNUSED
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_C
0003            # SymbolReference, value:SymbolTemperatur Pointer
01              # ColorIndex, value: OEM_GREEN4
000000          # UNUSED

                # Widget instance number:5
                # Word 37
CA01            # A661_CMD_CREATE
2C              # COMMAND SIZE (=44 bytes, words 37 – 47)
A160            # WIDGET TYPE (A661_LABEL)
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
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
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
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
13              # Alignment, value:A661_LEFT
81430000        # LabelString, value: "°C"

                # Widget instance number:7
                # Word 59
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            # FocusIndex, value:0
9878            # PictureReference, value:SymbolArrowUp
0000            # MaxStringLength, value:0
15              # PicturePosition, value:A661_CENTER
00              # AutomaticFocusMotion, value:A661_FALSE
0000            # UNUSED
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            # FocusIndex, value:0
987C            # PictureReference, value:SymbolArrowDown
0000            # MaxStringLength, value:0
15              # PicturePosition, value:A661_CENTER
00              # AutomaticFocusMotion, value:A661_FALSE
0000            # UNUSED
00000000        # LabelString, value: ""

                # Word 81
C0              # A661_END_LAYER_BLOCK
54E367          # CRC
# END DF
```

c-1

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C.4 Example of XML the Form of the Definition File

For the application illustrated in Section C.3, the corresponding XML form of the DF would be as follows. Note that parameters not mentioned take on the CDS-defined default value. Parent relationships are implied by the file structure. Relationship to the containing layer is not shown.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE A661_METAFILE SYSTEM « ../dtd/A661DF.dtd »>
<A661_METAFILE>
    <Panel
        Name = “CabinTempPanel”
        PosX = “11000”     PosY = “12700”
        SizeX = “7620”  SizeY = “5978”>
        <Picture
            Name = “TemperatureCelciusScale”
            PosX = “750”    PosY = “2390”
            SizeX = “6126” SizeY = “2490”
            PicutreRef = “SymbolTemperatureCelciusScale”
            />
        <Symbol
            Name = “TemperatureIndicatedPointer”
            PosX = “3810”    PosY = “2390”
            RotationAngle = “9100”
            StyleSet = “OEM_STYLESET_FREE_COLOR”
            ColorIndex = “OEM_WHITE”
            PictureReference = “SymbolTemperatureIndicatedPointer”
            />
        <Symbol
            Name = “TemperatureSelectedPointer”
            PosX = “3810”    PosY = “2390”
            RotationAngle = “1820”
            StyleSet = “OEM_STYLESET_FREE_COLOR”
            ColorIndex = “OEM_GREEN4”
            PictureReference = “SymbolTemperatureSelectedPointer”
            />
        <Label
            Name = “IndicatedTempDRO”
            PosX = “2864”  PosY = “1768”
            SizeX = “950” SizeY = “622”
            StyleSet = “OEM_STYLESET_NORMAL_READOUT”
            Font = “OEM_STYLESET_DEFAULT_FONT”
            MaxStringLength = “4”
            Alignment = “A661_RIGHT”
            LabelString = “24”
            />
        <Label
            Name = “IndicatedUnitLabel”
            PosX = “3984” PosY = “1768”
            SizeX = “473”  SizeY = “622”
            StyleSet = “OEM_STYLESET_NORMAL_READOUT”
            Font = “OEM_STYLESET_DEFAULT_FONT”
            MaxStringLength = “2”
            LabelString = “°C”
            />
        <PicturePushButton
            Name = “IncreaseSelectTemp”
            PosX = “473”    PosY = “473”
            SizeX = “2864” SizeY = “950”
            PictureReference = “SymbolArrowUp”
            />
        <PicturePushButton
            Name = “DecreaseSelectTemp”
            PosX = “3984”    PosY = “473”
            SizeX = “2864” SizeY = “950”
            PictureReference = “SymbolArrowDown”
            />
    </Panel>
</A661_METAFILE>
```

APPENDIX D
EXAMPLE OF "IN/OUT" WIDGET MANAGEMENT USING STYLESET PARAMETER

The inner states of an interactive widget are generally managed by the CDS upon crew member interaction. When the CDS changes the state of a widget, it sends an event to inform the UA of the interaction. In this case, the widget is considered as an IN widget.

If the UA wants to display its functional context though the interactive widget, the widget is considered as an OUT widget. The widget is used to provide functional information to the crew member. This information should be acknowledged by the StyleSet parameter.

Figure D-1illustrates an example of In/Out CheckButton management. The CDS manages a graphical feedback through the inner state, while the UA manages a functional feedback through the StyleSet parameter.

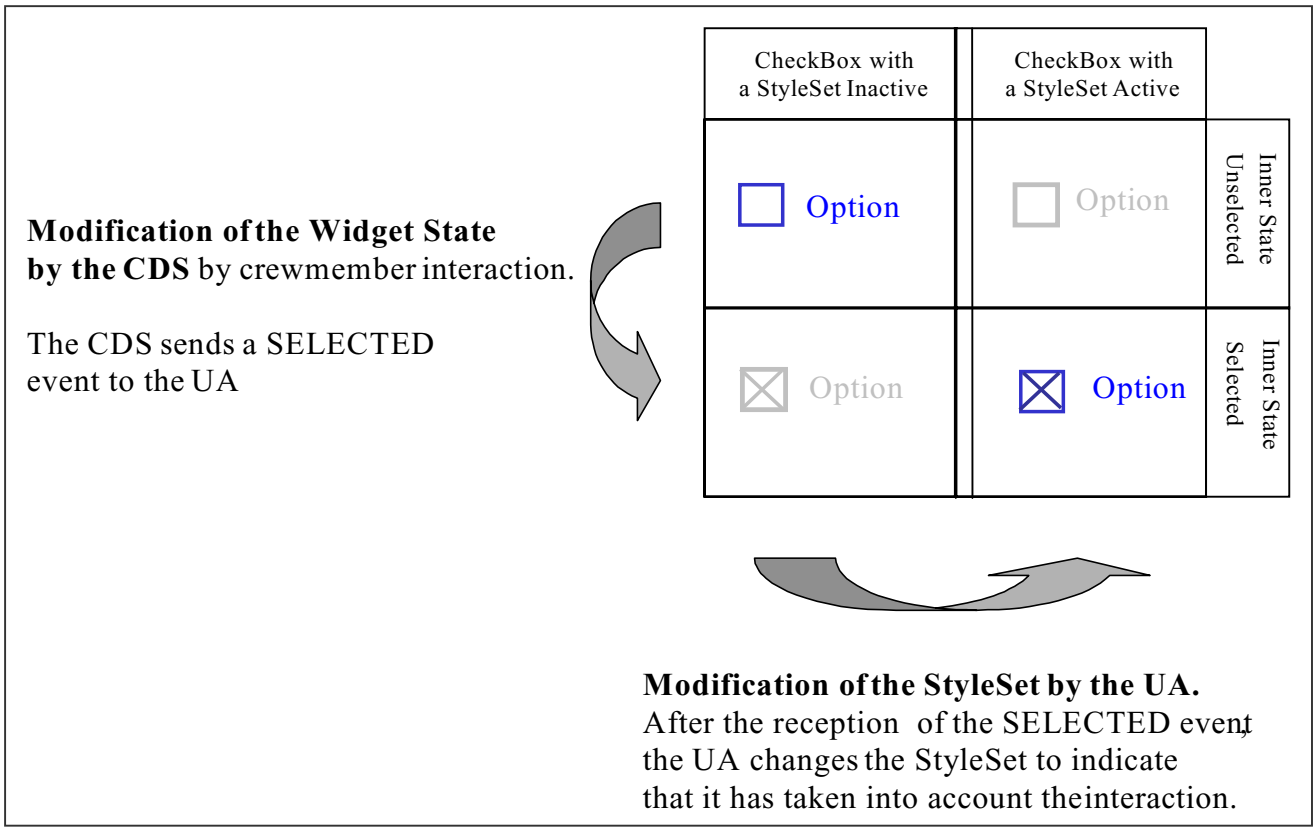


Figure D-1 "In/Out" Widget Management

APPENDIX E
MAP MANAGEMENT TUTORIAL

E.1 Examples of parameters definition for Map Management

a. ND as is the master application -

The ND has two basic mode 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).

b. The 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,

then the CDS is able to put the FMS Map data on the display.

c. The TCAS as the master application – c-1

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.

d. 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.

APPENDIX E
MAP MANAGEMENT TURORIAL

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.

APPENDIX E
MAP MANAGEMENT TUTORIAL

E.2.3 Removing Map Items

A specific ItemType is A661_NOT_USED. The parameter list for thie 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 correspond to different functional sets, for which it should define a different widget (MapHorz_ItemList).

However, the drawing priority of one Item is defined by the ItemIndex of this Item. Nevertheless, the ItemIndex of one Item is independent og the transmitting order of this 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 TURORIAL

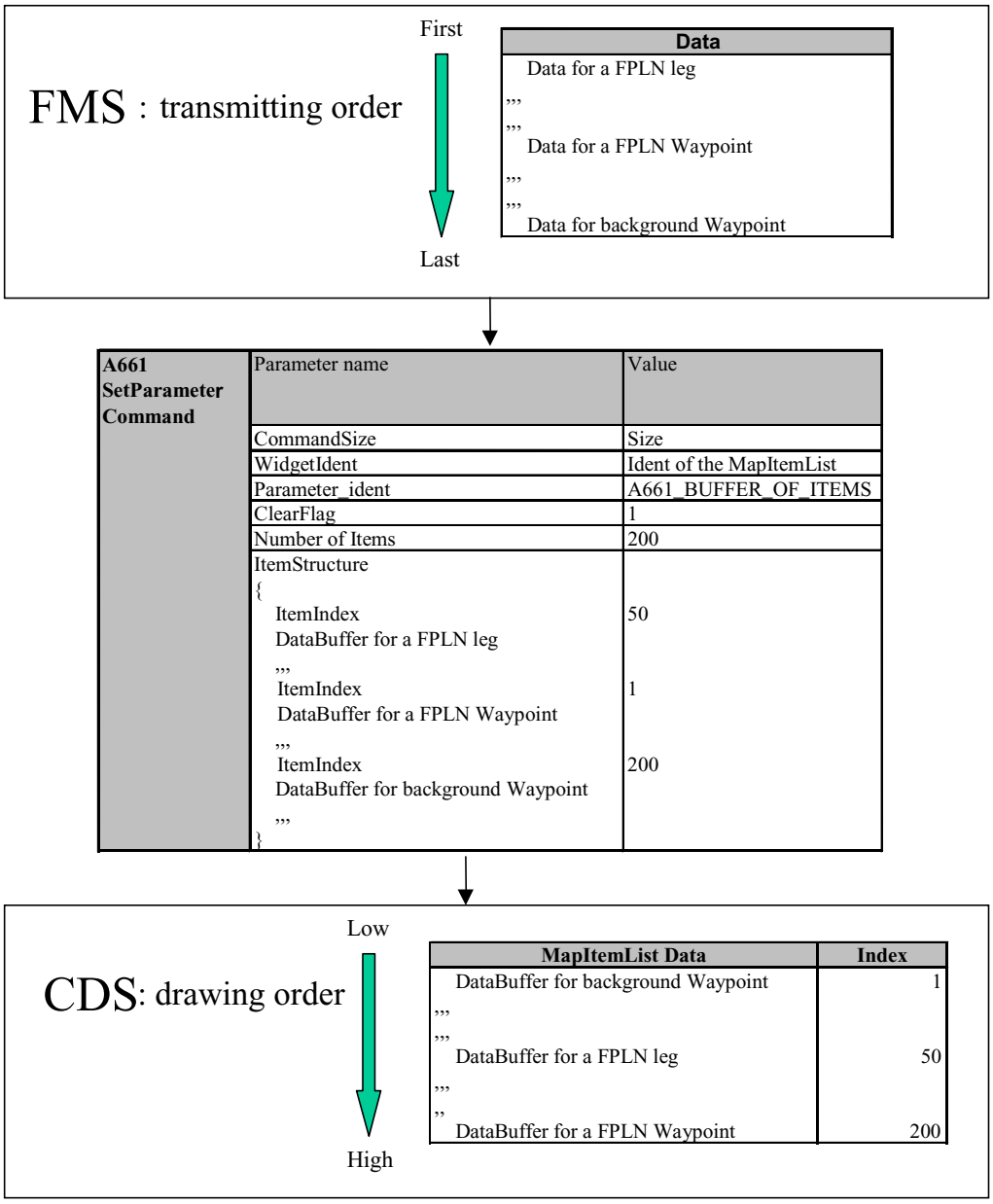


Figure E.4-1 Example of Dynamic Priority Management

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7465 USA

ARINC SPECIFICATION 661-1

COCKPIT DISPLAY SYSTEM INTERFACE

TO USER SYSTEMS

Published: June 26, 2003

This document is based on material submitted by various participants during the drafting process. Neither AEEC nor ARINC has made any determination whether there materials could be subject to claims of patent or other proprietary rights by third parties, and no representation or warranty, express or implied, is made in this regard. Any use of or reliance on this document should constitute an acceptance hereof "as is" and be subject to this disclaimer.

This is a working paper prepared for AEEC. It does not constitute air transport industry or ARINC approved policy, nor is it endorsed by the U.S. Federal Government, any of its agencies or others who may have participated in its preparation.

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

The first part of this document, printed on goldenrod-colored paper, contains descriptions of the changes introduced in Specification 661 by this Supplement. The second part consists of replacement material, organized by section number, for the Specification to reflect these changes. The modified and added material is identified by the "c-1" symbol in the margins. ARINC Specification 661 was updated by incorporating the replacement material. The goldenrod-colored pages are inserted inside the rear cover of the Specification.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete tabulation of the changes and additions to Specification 661 introduced by this Supplement. Each change or addition is defined by the section number and the title that will be employed when this Supplement is incorporated. In each case, a brief description of the change or addition 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.

ARINC Standard – Errata Report

1. Document Title

ARINC Specification 661-1: *Cockpit Display System Interfaces*
Published: June 26, 2003

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

(State why the correction is necessary.)

6. Submitter (Optional)

(Name, organization, contact information, e.g., phone, email address.)

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.

Please return comments to fax +1 410-266-2047 or standards@arinc.com

ARINC IA Project Initiation/Modification (APIM) Guidelines for Submittal

(Date of Submittal _____)

1. ARINC Industry Activities Projects and Work Program

A project is established in order to accomplish a technical task approved by one or more of the committees (AEEC, AMC, FSEMC) Projects generally but not exclusively result in a new ARINC standard or modify an existing ARINC standard. All projects are typically approved on a calendar year basis. Any project extending beyond a single year will be reviewed annually before being re-authorized. The work program of Industry Activities (IA) consists of all projects authorized by AEEC, AMC, or FSEMC (The Committees) for the current calendar year.

The Committees establish a project after consideration of an ARINC Project Initiation/Modification (APIM) request. This document includes a template which has provisions for all of the information required by The Committees to determine the relative priority of the project in relation to the entire work program.

All recommendations to the committees to establish or reauthorize a project, whether originated by an airline or from the industry, should be prepared using the APIM template. Any field that cannot be filled in by the originator may be left blank for subsequent action.

2. Normal APIM Evaluation Process

Initiation of an APIM

All proposed projects must be formally initiated by filling in the APIM template. An APIM may be initiated by anyone in the airline community, e.g., airline, vendor, committee staff.

Staff Support

All proposed APIMs will be processed by committee staff. Each proposal will be numbered, logged, and evaluated for completeness. Proposals may be edited to present a style consistent with the committee evaluation process. For example, narrative sentences may be changed to bullet items, etc. When an APIM is complete, it will be forwarded to the appropriate Committee for evaluation.

The committee staff will track all ongoing projects and prepare annual reports on progress.

Committee Evaluation and Acceptance or Rejection

The annual work program for each Committee is normally established at its annual meeting. Additional work tasks may be evaluated at other meetings held during the year. Each committee (i.e., AMC, AEEC, FSEMC) has its own schedule of annual and interim meetings.

The committee staff will endeavor to process APIMs and present them to the appropriate Committee at its next available meeting. The Committee will then evaluate the proposal. Evaluation criteria will include:

- Airline support – number and strength of airline support for the project, including whether or not an airline chairman has been identified
- Issues – what technical, programmatic, or competitive issues are addressed by the project, what problem will be solved
- Schedule – what regulatory, aircraft development or modification, airline equipment upgrade, or other projected events drive the urgency for this project

Accepted proposals will be assigned to a subcommittee for action with one of two priorities:

- High Priority – technical solution needed as rapidly as possible
- Routine Priority – technical solution to proceed at a normal pace

Proposals may have designated coordination with other groups. This means that the final work must be coordinated with the designated group(s) prior to submittal for adoption consideration.

Proposals that are not accepted may be classified as follows:

- Deferred for later consideration - the project is not deemed of sufficient urgency to be placed on the current calendar of activities but will be reconsidered at a later date
- Deferred to a subcommittee for refinement – the subcommittee will be requested to, for example, gain stronger airline support or resolve architectural issues
- Rejected – the proposal is not seen as being appropriate, e.g., out of scope of the committee

3. APIM Template

The following is an annotated outline for the APIM. Proposal initiators are requested to fill in all fields as completely as possible, replacing the italicized explanations in each section with information as available. Fields that cannot be completed may be left blank. When using the Word file version of the following template, update the header and footer to identify the project.

ARINC IA Project Initiation/Modification (APIM)

Name of proposed project **APIM #: _____**

Name for proposed project.

Suggested Subcommittee assignment

Identify an existing group that has the expertise to successfully complete the project. If no such group is known to exist, a recommendation to form a new group may be made.

Project Scope

Describe the scope of the project clearly and concisely. The scope should describe "what" will be done, i.e., the technical boundaries of the project. Example: "This project will standardize a protocol for the control of printers. The protocol will be independent of the underlying data stream or page description language but will be usable by all classes of printers."

Project Benefit

Describe the purpose and benefit of the project. This section should describe "why" the project should be done. Describe how the new standard will improve competition among vendors, giving airlines freedom of choice. This section provides justification for the allocation of both IA and airline resources. Example: "Currently each class of printers implements its own proprietary protocol for the transfer of a print job. In order to provide access to the cockpit printer from several different avionics sources, a single protocol is needed. The protocol will permit automatic determination of printer type and configuration to provide for growth and product differentiation."

Airlines supporting effort

Name, airline, and contact information for proposed chairman, lead airline, list of airlines expressing interest in working on the project (supporting airlines), and list of airlines expressing interest but unable to support (sponsoring airlines). It is important for airline support to be gained prior to submittal. Other organizations, such as airframe manufacturers, avionics vendors, etc. supporting the effort should also be listed.

Issues to be worked

Describe the major issues to be addressed by the proposed ARINC standard.

Recommended Coordination with other groups

Draft documents may have impact on the work of groups other than the originating group. The APIM writer or, subsequently, The Committee may identify other groups which must be given the opportunity to review and comment upon mature draft documents.

Projects/programs supported by work

If the timetable for this work is driven by a new airplane type, major avionics overhaul, regulatory mandate, etc., that information should be placed in this section. This information is a key factor in assessing the priority of this proposed task against all other tasks competing for subcommittee meeting time and other resources.

Timetable for projects/programs

Identify when the new ARINC standard is needed (month/year).

Documents to be produced and date of expected result

The name and number (if already assigned) of the proposed ARINC standard to be either newly produced or modified.

Comments

Anything else deemed useful to the committees for prioritization of this work.

Meetings

The following table identifies the number of meetings and proposed meeting days needed to produce the documents described above.

Activity	Mtgs	Mtg-Days
Document a	*# of mtgs*	*# of mtg days*
Document b	*# of mtgs*	*# of mtg days*

For IA staff use

IA staff assigned: _____

Forward to committee(s) (AEEC, AMC, FSEMC): _____

Potential impact: ____
(A. Safety B. Regulatory C. New aircraft/system D. Other)

Committee resolution: ____
(1. Authorized 2. Deferred 3. More detail needed 4. Rejected)

Assigned Priority: ____

A. – High (execute first) B. – Normal (may be deferred for A.)