ARINC

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS PART 1 AVIONICS INTERFACES, BASIC SYMBOLOGY, AND BEHAVIOR

ARINC SPECIFICATION 661P1-7

PUBLISHED: June 17, 2019

Prepared by the AEEC
Published by
SAE ITC
16701 Melford Blvd., Suite 120, Bowie, Maryland 20715 USA

SAE ITC
An SAE International Affiliate

DISCLAIMER

THIS DOCUMENT IS BASED ON MATERIAL SUBMITTED BY VARIOUS PARTICIPANTS DURING THE DRAFTING PROCESS. NEITHER AEEC, AMC, FSEMC NOR SAE ITC 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.

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

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

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

ARINC SPECIFICATION 661P1-7

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS PART 1 AVIONICS INTERFACES, BASIC SYMBOLOGY, AND BEHAVIOR

Published: June 17, 2019

Prepared by the Airlines Electronic Engineering Committee (AEEC)
Adopted by the AEEC Executive Committee

	Adoption Date	Published Date
Specification 661	November 12, 2001	April 22, 2002
Supplements to this ARINC Standard		
Specification 661-1	March 6, 2003	June 26, 2003
Specification 661-2	October 27, 2004	June 30, 2005
Specification 661-3	September 26, 2007	November 15, 2007
Specification 661-4	March 31, 2010	May 10, 2010
Specification 661-5	April 24, 2013	July 1, 2013
Specification 661-6	April 26, 2016	September 1, 2016
Specification 661P1-7	May 1, 2019	June 17, 2019

A summary of the changes introduced by each supplement is included at the end of this document.

FOREWORD

The AEEC, SAE ITC, and ARINC Standards

ARINC Industry Activities, an SAE ITC program, organizes aviation industry committees and participates in related industry activities that benefit aviation at large by providing technical leadership and guidance. These activities directly support aviation industry goals: promote safety, efficiency, regularity, and cost-effectiveness in aircraft operations.

ARINC Industry Activities organizes and provides the secretariat for international aviation organizations (AEEC, AMC, FSEMC) which coordinate the work of aviation industry technical professionals and lead the development of technical standards for airborne electronic equipment, aircraft maintenance equipment and practices, and flight simulator equipment used in commercial, military, and business aviation. The AEEC, AMC, and FSEMC develop consensus-based, voluntary standards that are published by SAE ITC and are known as ARINC Standards. The use of ARINC Standards results in substantial technical and economic benefit to the aviation industry.

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 organization 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 document:

An Errata Report solicits any corrections to existing text or diagrams that may be included in a future Supplement to this ARINC Standard.

An ARINC IA Project Initiation/Modification (APIM) form solicits any proposals for the addition of technical material to this ARINC Standard.

APPENDICES

1.0 INTRODUCTION

1.1 Purpose and Scope

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

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

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

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

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

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

1.2 Relationship to Other Documents

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

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

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

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

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

ARINC Characteristic 739A: ***Multi-Purpose Control and Display Unit***

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

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

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

ARINC Specification 664: *Aircraft Data Network*

1.3 Interoperability

1.3.1 General

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

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

COMMENTARY

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

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

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

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

1.3.2 Interface Standards

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

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

COMMENTARY

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

1.3.3 Modularity

Architecturally, the CDS should be an integrated system comprised of modular hardware and software components. It should be possible to include optional

features to individual units, as determined by airline user requirements, with minimum impact on the existing functions.

This specification emphasizes that software necessary to add or change display functionality of the display system should be contained within the User Application (e.g., Flight Management System (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 is 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 Applicability

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

Display types include, but are not limited to:

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

1.9 Associated Electronic Files

ARINC Specification 661-7 contains both this document (PDF) and electronic eXtensible Markup Language (XML) support files. The copyright and disclaimer contained in this PDF also apply to any and all electronic support files.

Table 1.9 – Electronic Support Files Associated With This Document

File	Description/Document Section/Location URL
df.zip	Contains the Definition File schema with examples
	Section 6.1.1
	https://www.aviation-ia.com/support-files/661-7-df
look.zip	Contains the Look Capacities and Look Definition schemas with examples
	Appendix J
	https://www.aviation-ia.com/support-files/661-7-look
uacdsinterface.zip	Contains the UA/CDS Interface Capacities and UA/CDS Interface Definition schemas with examples
	Appendix K
	https://www.aviation-ia.com/support-files/661-7-uacdsinterface
charmap.zip	Contains the Character Set Map schemas with examples
	Appendix L
	https://www.aviation-ia.com/support-files/661-7-charmap

Some information is uniquely specified in this document. Some information is contained only in the electronic files. The information contained in this document and the electronic support files is intended to be consistent. In case of discrepancies between this document and the electronic support files, please notify ARINC Industry Activities.

1.9.1 XML Schema Files (XSD) Associated with this Document

All XSD schemas have a unique identifying target namespace URI, of the form:

`http://www.aviation-ia.com/support-files/661-7-SCHEMA/VERSION`

where the VERSION part of the URI will change if modifications are published. Target namespaces are identifiers and do not infer a file location.

Some schemas are constructed from sub-schemas (in separate files) and aggregated together into a principal schema via inclusion. These schemas may share a common target namespace URI.

By convention, XML documents referring to a schema uniquely identify the file via its target namespace URI *and* filename e.g.,

```
xs:schemaLocation="http://www.aviation-ia.com/support-files/661-7-df/1 a661.xsd"
```

For XML Validation to supplied schemas it is expected that users of this document use a local copy (or their own host), combined with a catalogue mechanism to redirect the xsi:schemaLocation information to their local unzipped copy URL. This convention and mechanism ensures XML files remain portable.

Table 1.9.1 – XSD Schemas, URI, Version and Location

Schema File/URI (With Version)	Support File
a661.xsd *https://www.aviation-ia.com/support-files/661-7-df/1*	df.zip
Look.xsd *https://www.aviation-ia.com/support-files/661-7-look/2*	look.zip
LookCapacities.xsd *https://www.aviation-ia.com/support-files/661-7-look/2*	
LookDefinition.xsd *https://www.aviation-ia.com/support-files/661-7-look/2*	
LookGeneral.xsd *https://www.aviation-ia.com/support-files/661-7-look/2*	
UACDSInterface.xsd *https://www.aviation-ia.com/support-files/661-7-uacdsinterface/1*	uacdsinterface.zip
UACDSInterfaceCapacities.xsd *https://www.aviation-ia.com/support-files/661-7-uacdsinterface/1*	
UACDSInterfaceDefinition.xsd *https://www.aviation-ia.com/support-files/661-7-uacdsinterface/1*	
UACDSInterfaceGeneral.xsd *https://www.aviation-ia.com/support-files/661-7-uacdsinterface/1*	
charmap.xsd *https://www.aviation-ia.com/support-files/661-7-charmap/1*	charmap.zip

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 using cursor control devices, keyboards, touch screens, scrolls wheels, and other input devices. 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., 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 Files (DF) the characteristics of all the instances of each widget it uses in the initial design or expects to use. This is described in detail in Section 4.1.1, Definition File and User Application Layer Definition (UALD). These widgets are allocated inside the CDS.

The UA addresses its widgets through a run-time protocol. This is described in detail in Section 4.2, Run-time Communications. The UA animates the display format by setting the accessible parameters of the widgets in order to reflect its functional context. The run-time protocol also allows the CDS to report 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 term "look" refers to the appearance of widgets, for example, graphical characteristics such as color and border properties. The term "feel" refers to the behavior of widgets, for example, the manner in which an interactive widget reacts to being selected by a crew member.

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.

However, 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 the possibility to create a common look across all aircraft, and common implementation of behavior consistent with that airframe manufacturer’s cockpit philosophy. The style guide defined by the airframe manufacturer describes the “look and feel” inside the cockpit, and thus provides the necessary information to UAs for their HMI interface design.

There are two categories of widget interface definition: (1) specification or compile-time information, and (2) run-time interface:

- 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 2.2-1 illustrates ARINC 661 protocol principles applied in a typical CDS system architecture. Higher values in the stacks take precedence over the lower modifiable values.

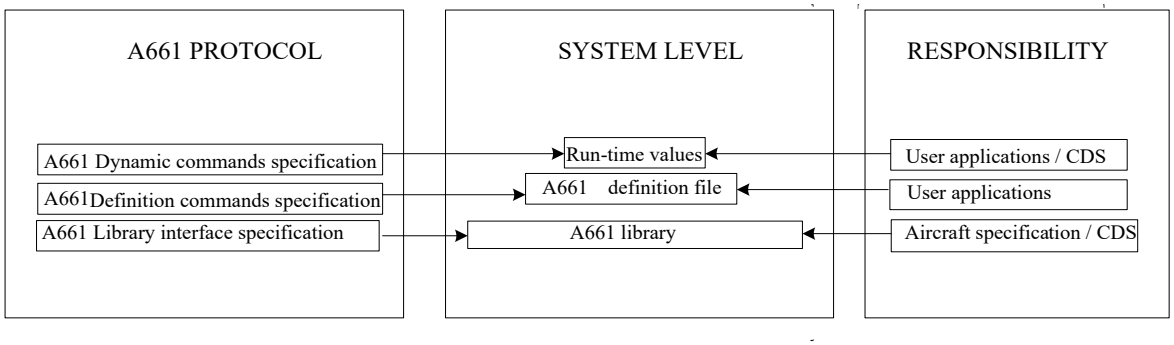


Figure 2.2-1 – ARINC 661 Protocol Principles

2.2.1 Definition Phase

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

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

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

The DF describes widget hierarchical structures.

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

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

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

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

COMMENTARY

Defining the end of the definition phase and the beginning of the run-time phase is beyond the scope of this document. It is the 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:

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

From CDS to UA:

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

2.2.3 Special Conditions

2.2.3.1 Initialization

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

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

2.2.3.2 Need for Re-initialization

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

2.2.3.3 Suppression of a Layer from Display

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

2.2.4 ARINC 661 Conformance

A CDS conforms to ARINC 661 when: (a) it implements the prescribed CDS-UA communication mechanism (Section 4.0) including DF files and run-time messaging model, and (b) all widgets and widget parameters/capabilities that the CDS does implement are compliant with this standard.

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

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

2.2.5 ARINC 661 Library Evolution

It is expected that ARINC Specification 661 will evolve over time. The guidelines presented in Appendix G should be used when proposing changes and additions to the Widget Library. The guidelines address important considerations for evaluating changes.

2.2.5.1 Deprecation of ARINC 661 Widgets and Features

One method used for modifying ARINC Specification 661 is called “Deprecation.” Using this method, changes can be made to ARINC 661 without causing earlier implementations to become immediately obsolete. Portions of this document designated as having been “Deprecated” are deemed to be out of date. Deprecated material will eventually be removed from this document in a future Supplement to this standard. Additionally, new material may be added to ARINC 661 to replace Deprecated material. Implementers should continue to support deprecated features while they remain in the document.

The rationale for each deprecation can be found in the change pages (the “ABC” pages) published with the associated supplement found at the end of this document.

2.2.5.2 Widget Extension

Another method for modifying ARINC Specification 661 is called Widget Extension. This method modifies ARINC 661 by adding optional functionality to existing widgets. Parameters needed for this new function are specified in the definition file in an extension block. There may also be runtime modifiable parameters and events associated with a Widget Extension. See Section 8.0 for details on available extensions.

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, 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.

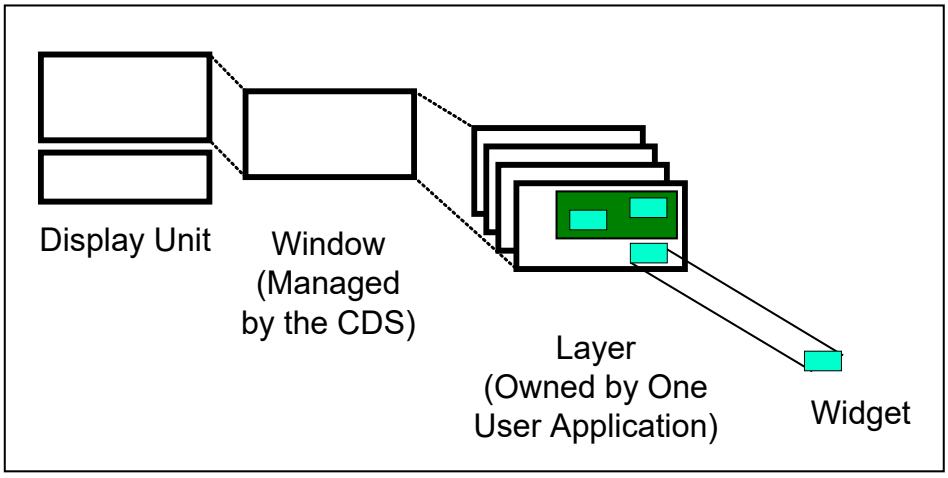


Figure 2.3 – Window/Layer Illustration

2.3.1 Window Definition

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

A format image rendered to a display unit surface is constructed from one or more windows.

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

2.3.2 Layer Definition

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

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

COMMENTARY

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

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

2.3.2.1 Layer Graphical Definition

An ARINC 661 layer graphical definition has the following characteristics:

It is a graphical layer inside a window

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

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

2.3.2.2 Layer Content Management

Layer content management has the following characteristics:

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

COMMENTARY

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

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

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

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

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

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

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

2.3.2.3 Layer Priority Management

Layer content management has the following characteristics:

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

Popup widgets (e.g., PopUpMenu and PopUpPanel) and popup parts of other widgets should always be drawn on top of all other widgets in the window.

Several popups can be displayed at the same time. In this case the graphical priority of the popup is implementation dependent. Graphical priority could be based on StyleSet, UADF order of the widget inside the UALD, or temporality.

Additionally, popup can be clipped by ancestor container(s), as part of the implementation dependent look and feel.

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, Notification from CDS to UA.

2.3.2.4.1 Visibility

The layer visibility is managed by the UA through a visibility parameter (an exception to this is when a NoServiceMonitor widget is present in the layer; see Section 3.8.10, NoServiceMonitor).

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

2.3.2.4.2 Activity

The activity of the layer is controlled by the CDS. When a layer is active, the CDS should update the data from the UA that owns the layer, even if the layer is not visible. Refer to Section 4.4.3.1, Request from UA to CDS, for A661_REQ_LAYER_ACTIVE.

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

The CDS sends the A661_NOTE_CYCLIC_ACTIVATION notification to the UA at a periodic rate while the layer is active. The CDS sends a A661_NOTE_LAYER_IS_INACTIVE notification to the UA when CDS de-activates the layer.

When a layer becomes inactive, its visibility is turned off by the CDS (an exception to this is when a NoServiceMonitor widget is present in the layer; see Section 3.8.10, NoServiceMonitor). 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 or an

A661_NOTE_REINIT_LAYER_DATA is received, the UA should reinitialize the layer's data.

COMMENTARY

The specific use of the Activity/Inactivity property on a layer of the CDS is outside the scope of this standard. This should be defined by the airframe manufacturer.

When the layer is inactive, only the A661_REQ_LAYER_ACTIVE request should be processed for that layer.

It is implementation dependent whether A661_REQ_LAYER_ACTIVE or A661_REQ_LAYER_INACTIVE requests from the UA will be fulfilled. For example, the CDS may activate a layer if and only if it is not hidden by its ancestors (See Appendix H).

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).

The context number allows the UA to detect race conditions with the CDS and determine the state of the CDS at the time it sent an event to the UA. See Section 3.1.2.2 for more discussion on race conditions.

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 4.4.3.1, ARINC 661 Request from UA to CDS, forA661_REQ_LAYER_ACTIVE.

2.3.4 Positioning and Size Within Window

Figure 2.3.4 illustrates graphic references for widget positioning.

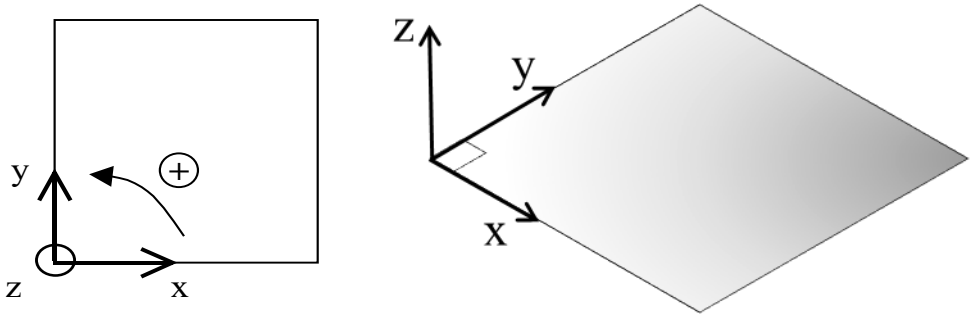


Figure 2.3.4 – Graphic References for Widget Positioning

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.

The X-axis is pointing right, the Y-axis is pointing up, the Z-axis is pointing in the same direction as the cross product of the vector x and vector y (toward the reader).

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

2.3.4.2 Angles

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

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

2.3.4.3 Screen Units of Measurements

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

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

2.3.5 Cursor Management

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

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

- An interactive widget is considered to have Focus when it will receive device input triggered by a crew member through non-CCD devices (such as keyboard).
- An interactive widget is Highlighted when the cursor passes over its interactive area. Depending on the implementation, the click may select and/or focus the widget.

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

When input devices not described above are used, such as a touch screen, the use of cursor, Focus, and Highlight is implementation dependent (i.e., whether or not touch screen inputs affect cursor position). A CDS may choose to apply Highlight behavior to touch screen interactions without moving the cursor shape over the interactive area of the widget being touched. It is also possible to have a system with no cursor.

2.3.5.1 From UA to CDS

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

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

2.3.5.2 From CDS to UA

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

COMMENTARY

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

- Interactivity features
- Movement rules between DUs
- Movement rules between windows
- Link between the cursor shape and the window characteristics, for example frozen window
- Response to cursor inputs involving overlapping widgets
- Behavior of cursor due to touch screen or other input device actions

Determination of all cases where cursor snaps might occur is implementation dependent. For example, when the cursor is on a widget owned by a UA that

suddenly fails, the placement of the cursor should be defined. The cursor could go to another widget in such a case. Another case to consider is where to put the cursor when a window or layer is first initialized and displayed. The definition and use of default locations for such cases should be considered.

Precise response time requirements depend on user system operational requirements. Table 2.3.5.2 provides guidelines that should be considered by system designers in determining computer processing requirements and software architecture necessary to support this interface. The same guidelines should be considered for touch screen timing as well, although touch interfaces may have more stringent latency requirements due to the co-location of the input device (i.e., finger) with the cursor (if applicable) and/or interactive widget.

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

Table 2.3.5.2 – Guidelines for Cursor-Control Timing

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

COMMENTARY

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

COMMENTARY

In cases where the active areas of two interactive widgets overlap, if the pilot makes a selection in the overlap area, it is recommended the CDS send the applicable event associated with the widget on top. A

widget is defined to be on top of another widget if the widget is listed after the other widget in the DF and shares at least a portion of the display space with the other widget. This may not be applicable to all widget types (e.g., CursorOver).

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

2.3.6 Touch Screen Management

Raw touch screen inputs are managed by the CDS. Some widgets may be specifically designed for the CDS to communicate touch interactions with UAs. For example, TouchArea is intended to be used when the CDS passes basic touch position and state data for the UA to interpret, while GestureArea is intended to be used for the CDS to interpret the touch position and state information and send resulting gesture-based events to the UA. In all cases, the CDS is responsible for translating touch screen inputs into screen coordinates and states for the supported number of touch points.

In addition to the touch specific widgets, other interactive widgets (e.g., PushButton, Slider, etc.) may respond to touch screen inputs in a similar manner to other input devices. The mapping of touch actions to interactive widget events is implementation dependent.

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 this section, Widget Library.

3.1.1 Widget Identification

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

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

Therefore, the interface between CDS and UA uniquely defines widgets by the combination: [UserApplicationIdent].[LayerIdent].[WidgetIdent].

COMMENTARY

[UserApplicationIdent] is resolved by the CDS using information about the system architecture, in particular, knowledge about the physical and logical interfaces between UAs and the CDS. This information is outside the scope of ARINC 661, and [UserApplicationIdent] does not appear in the ARINC 661 protocol. This may be, but is not necessarily, the same identifier as defined in the AppliIdent field of the User Application Definition File (UADF).

3.1.2 Widget States

3.1.2.1 Widget States Definition

Four different levels, illustrated in Figure 3.1.2.1 define widget states:

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

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

- Visible

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

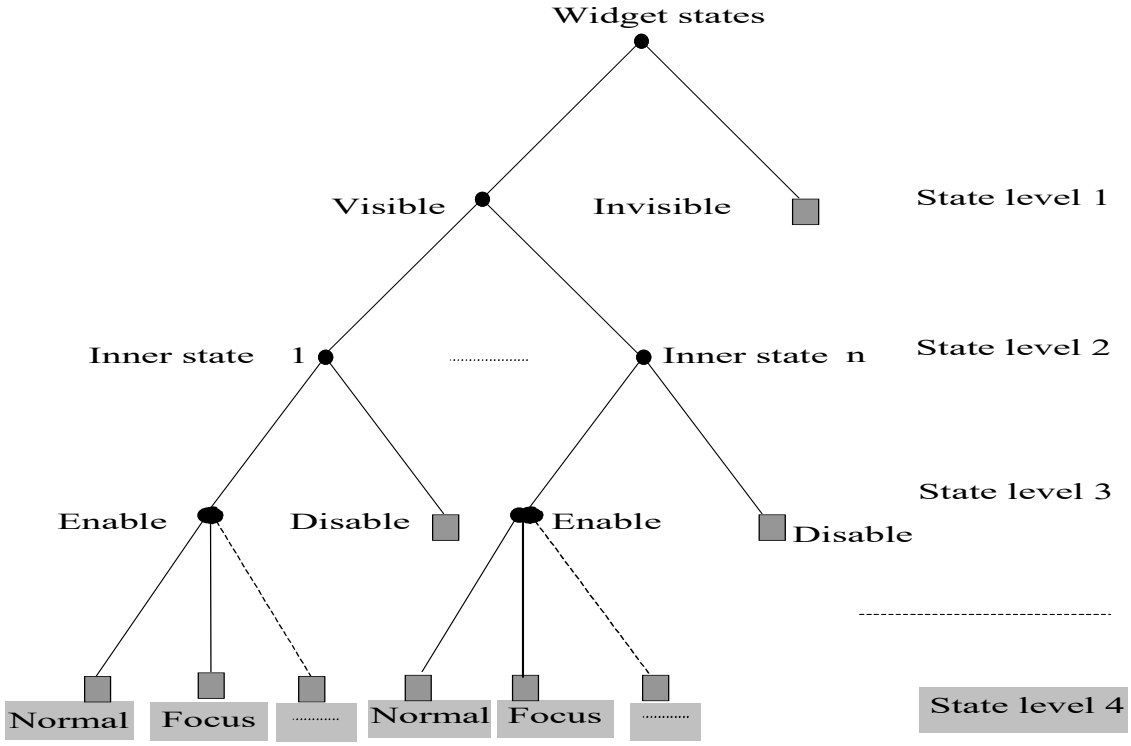


Figure 3.1.2.1 – Example of Widget States Levels

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

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

3.1.2.2 Inner State Management: “Race Condition”

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

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

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

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

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

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

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

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

- The race condition can be mitigated to some degree by minimizing the update of IN/OUT widget by the UA. Care must be taken to ensure that the CDS inner state of the widget is consistent with the UA state of the widget.
- For “EditBox” widgets, the race could also be minimized by the UA using the EditBox_Open and EditBox_Close event to restrict sending A661_String to the Widget while it is actively being edited, but a race condition can still occur between EditBox_Open and a SetParameter.
- The UA implementation can also use the UA layer’s Context Number to help interpret the event, particularly for cases where pilots are selecting from a periodically updated list.
- For example, the UA sends a ComboBox EntryList containing “A”, “B”, “C”, and “D” in the first frame and “A”, “B”, “Q”, and “D” in the second frame to the CDS. The CDS sends an EntryNumber = 3 to the UA indicating the third item in the list has been selected. It is ambiguous to the UA as to whether the EntryNumber corresponds to the first (i.e., “C”) or second (i.e., “Q”) EntryList. Now, if the UA layer’s Context Number for the first EntryList = 125 and the

second EntryList = 126, and the CDS echoed Context Number for the EntryNumber event = 126, the UA knows the selection was “Q”.

- As of Supplement 3, UA Validation can be used to help address the race condition. See Section 2.3.2.5.

Also, a human factors race condition occurs if the UA changes the characteristics of a widget just before a user interacts with it. HMI design or widget implementations could be used to address this issue.

3.1.3 Commonly Used Parameters

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

3.1.3.1 Identification of the Widget

Widget Identification Parameters are defined in Table 3.1.3.1.

Table 3.1.3.1 – Widget Identification Parameters

Parameter	Description
WidgetType	Type of widget.
WidgetIdent	Identifier of the widget (refer to Section 3.1.1,) WidgetIdent is a non-null positive value ([WidgetIdent] >0).
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
Visible	A661_FALSE: The widget will not be rendered (invisible). A661_TRUE: If all its ancestors are visible, the widget will be rendered. If any one of its ancestors is invisible, the widget will not be rendered, whatever the value of its visible parameter.
Enable	A661_FALSE: The widget will **be disabled (**not interactive). A661_TRUE or A661_TRUE_WITH_VALIDATION: If all its ancestors are enabled, the widget will be interactive. If any one of its ancestors are disabled, the widget will **also be disabled**, whatever the value of its Enable parameter. An invisible widget is not interactive, independent of the value of its Enable parameter.
Anonymous	A661_FALSE: run-time accessible. Widget run-time parameters can be modified by the UA. A661_TRUE: anonymous. Widget run-time parameters cannot be modified by the UA.

3.1.3.3 Look and Feel Characteristics of a Widget/Symbol:

Widget State Parameters are defined in Table 3.1.3.3.

Table 3.1.3.3 – Widget StyleSet Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>StyleSet</td><td>StyleSet allows the UA to select from a predefined set of graphical characteristics to be applied to a widget. This serves two purposes. First, many graphical capabilities (color depth, halo, fill styles, line weights/patterns, blinking, transparency, fonts, character highlighting, kerning, rotation, etc.) are inherently a function of CDS architecture. Guidance requiring or disallowing any of these characteristics is beyond the scope of this document.

Second, the application of these characteristics is usually intended by the aircraft OEM to be consistent across all UAs for common state conditions. Indexing among predefined styles supports this goal. Common state conditions can be defined as conditions that impact more than one user application in the same way (e.g., Alert, Caution). It can also be used by a single application for convenience and control of hidden characteristics.

Thus, any graphical characteristics set by StyleSet that match individually accessible graphical characteristics will be overridden by the values specified 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.

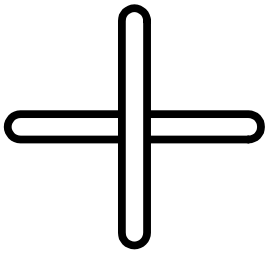
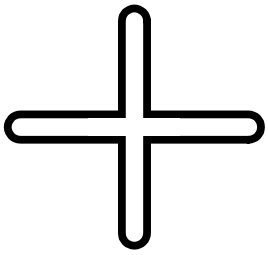
This specification defines one default StyleSet value:
STYLE_SET_DEFAULT
meaning that default graphical characteristics will be used.

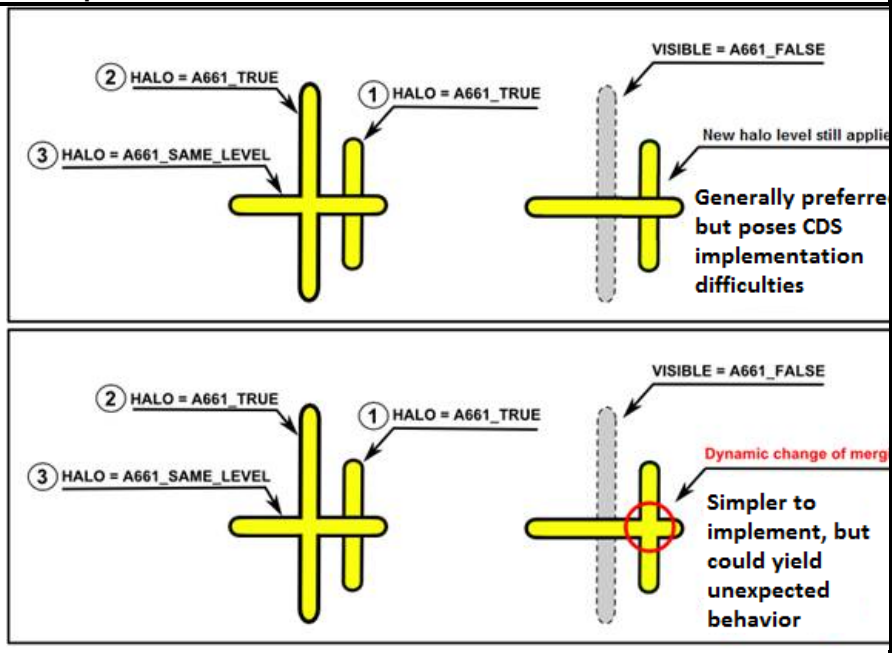
Aircraft OEM (or CDS supplier) defines the list of StyleSet values.

Examples of possible StyleSet values:
<table>
<tr><td>STYLE_SET_NOMINAL</td><td>STYLE_SET_SELECTED</td></tr>
<tr><td>STYLE_SET_ADVISORY</td><td>STYLE_SET_PRESELECTED</td></tr>
<tr><td>STYLE_SET_CAUTION</td><td>STYLE_SET_ENGAGED</td></tr>
<tr><td>STYLE_SET_WARNING</td><td>STYLE_SET_ARMED</td></tr>
<tr><td></td><td>STYLE_SET_NOT_ENGAGED</td></tr>
</table></td></tr>
<tr><td>Halo</td><td>Halo is a full outline of a Graphic primitive (Gp) widget in a contrasting color (typically black) to enhance readability.

A661_FALSE:
No halo is drawn.

A661_TRUE:
Halo is drawn on top of previously drawn Gp widget, that is, a new halo level is used. If the vectors of the Gp widget intersect another widget, the halo will cut through the vectors and halos drawn underneath. In the example figure below, the outline of the vertical GpLine widget with Halo set to A661_TRUE is drawn on top of the previously drawn horizontal GpLine widget.</td></tr>
</table>

Parameter	Description
	 A661_SAME_LEVEL: Halo is merged between all Gp widgets drawn in succession that have their Halo parameter set to A661_SAME_LEVEL, such that a continuous halo runs around the outside of intersecting vectors. In the example figure below, the outline of the vertical GpLine widget with Halo set to A661_SAME_LEVEL is drawn on top of the previously drawn horizontal GpLine widget.  The current halo level applies to all Gp widgets in the draw order until a widget set to True is reached, which ends the previous halo level. For example, if a GpLine with Halo parameter set to True is followed by a GpLine with Halo set to Same_Level, then a GpRectangle with Halo set to True, both GpLine widgets should have the same halo level, and the GpRectangle's halo should be displayed over the GpLine widgets. The rule applies to Connector widgets as well. The haloing of widgets in a connected layer should be handled with respect to the halo level of the layer owning the Connector widget. Therefore, if the first Gp widget in the connected layer has its Halo parameter set to Same_Level, its halo is treated as if it were in the layer owning the associated Connector widget. UA developers should be mindful that a connected layer from another system may alter the halo level, so it is good practice to always set the Halo parameter after the Connector widget accordingly. Same_Level is intended to be used by widgets that are consecutive in the draw order. Placing a widget without a Halo attribute in between a series of widgets with Halo set to Same_Level is not advised. It is implementation dependent how the presence of a widget without a Halo attribute impacts the current Halo level (this may also vary from widget to widget, since some complex widgets are themselves composed of several graphical objects). Similarly, the use of Same_Level by widgets implies a dependency on a prior widget whose Halo attribute is set to True. Therefore, making a widget whose Halo parameter is set to True not visible can impact how subsequent widgets set to Same_Level are displayed (see figure below).

Parameter	Description
	 It is advised that UA designers define visual priority and Halo settings to avoid situations illustrated in the above figure. It is recommended that for a sequence of widgets that share a halo level, the widgets that are conditionally visible should have their Halo attribute set to Same_Level. Halo value of True should be used for widgets that are always set to visible, or when subsequent widgets with a Halo of Same_Level share the same visibility conditions. It is implementation dependent how the presence of a non-visible widget with Halo set to True impacts the Halo level of subsequent widgets with Halo set to Same_Level.
StartCursor-Pos/StartCursor-PosByte	**Controls the editing position indicator (cursor) in a text field upon entering edit mode. The value is the index of the *editable character* in the displayed control. A value of 1 would place the cursor such that keyboard entry would overwrite (or insert at) the first editable character, for text read from left to right this would be the left-most editable character.** **Setting this parameter when mid-edit will have no effect until the next edit operation.** **The following behavior is implementation dependent and may also be configurable via the StyleSet parameter:** **a) The cursor form (e.g., I beam or block □),** **b) Interpretation of zero (e.g., as no-effect or highlight all).**

3.1.3.3.1 Predefined Look-Up Look and Feel Characteristics

The following widget/symbol characteristics are defined to provide common means to control look and feel attributes. In each case the provided "index" value corresponds to an entry into a "table" which provides the visual effect definition. These tables are pre-defined and static at run-time. The table contents, their definition mechanism, and whether or not they are configurable (before the run-time phase commences) is implementation dependent.

In each case StyleSet values may combine with these properties to affect how a widget is rendered.

Table 3.1.3.3.1 – Graphical Look-Up Parameters

Parameter	Description
ColorIndex	Index into a color palette e.g. defined as RGBA values.
FillIndex	Index to a fill style. A fill style may be a solid color fill, a patterned fill, an alpha blend, or other visual attribute.
FillStyleIndex	Index to a fill style used by Map Items. A fill style may be a solid color fill, a patterned fill, an alpha blend, or other visual attribute. Whether or not FillIndex and FillStyleIndex reference the same definitions is implementation dependent.
Font	Font values are considered to be an index to a table that alters how text is rendered. E.g., Font values may correspond to specific font family, appearance, point size or other textual attributes.
PictureReference\ AlternatePictureReference	Index to identify a picture. E.g., index into an array of pre-loaded bitmaps, or an index into an image-map or a reference to a PictureStructure provided as part of the DF (see Section 7.0).

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, SizeY parameters define a boundary area for the widget. Whether graphical characteristics outside this area are clipped is implementation dependent and may vary based on widget type. If clipping is applied for a widget, its clipping area applies to all of its descendent widgets as well.

This area defines the static area of the widget. For widgets that also contain a “Pop Up part” (e.g., ComboBox), only the static part is constrained by these parameters.

3.1.3.5 Parameters Related to Focus Navigation

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

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

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

Widget Common Structure is defined in Table 3.1.3.5.

Table 3.1.3.5 – Widget Common Structure

Parameter	Description
NextFocusedWidget	Widget ident of next widget to be focused. A value of zero (0) means there is no next widget specified.
AutomaticFocusMotion	A661_FALSE: No automatic motion: the focus remains on the widget until an explicit move of the focus. A661_TRUE: Allows automatic motion of the focus to the widget specified by the NextFocusedWidget parameter.

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

If a widget in the focus sequence is either invisible or disabled, the CDS skips over that widget in the order, and places focus on the next visible and enabled widget in the sequence.

The focus order does not inherently wrap around. If wrap-around behavior is desired, the NextFocusedWidget parameter of the last widget in the sequence should reference the first widget in the sequence.

The CDS may support the focus sequence in reverse order as well (e.g., for reverse tabbing). How this reverse order is defined in cases where there are multiple reverse paths is implementation dependent.

3.1.4 Widget Events

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

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

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

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

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

Widgets that trigger events are listed in Tables 3.1.4-1 and 3.1.4-2.

Table 3.1.4-1 – Widget Event Cross Reference

Widgets \ Events	A661_EVT_CURSOR_ENTER	A661_EVT_CURSOR_INSIDE	A661_EVT_CURSOR_EXIT	A661_EVT_CURSOR_POS_CHANGE	A661_EVT_EDITBOX_OPENED	A661_EVT_FIRST_VIS_ENTRY_CHANGE	A661_EVT_FRAME_POS_CHANGE	A661_EVT_GESTURE_DIR_FLICK	A661_EVT_GESTURE_DOUBLE_TAP	A661_EVT_GESTURE_DRAG	A661_EVT_GESTURE_FLICK	A661_EVT_GESTURE_HOLD	A661_EVT_GESTURE_PINCH	A661_EVT_GESTURE_PRESS_AND_HOLD	A661_EVT_GESTURE_ROTATE	A661_EVT_GESTURE_TAP	A661_EVT_INCREMENT	A661_EVT_ITEM_IN_BOUNDS	A661_EVT_ITEM_OUT_OF_BOUNDS	A661_EVT_ITEM_SYNCHRONIZATION	A661_EVT_KEY
ActiveArea																					
CheckButton																					
ComboBox																					
ComboBoxEdit					X																
CursorOver	X	X	X																		
CursorPosOverlay				X																	
EditBoxMasked					X																
EditBoxMultiLine					X																
EditBoxNumeric					X																
EditBoxNumericBCD					X																
EditBoxText					X																
GestureArea								X	X	X	X	X	X	X	X	X					
KeyboardArea																					X
ScrollWheelArea																	X				
MapHorz																				X	
MapHorz_Container																					
MapHorz_ItemList																		X	X		
MapHorz_Source																	X				
MapVert																				X	
MapVert_ItemList																					
MapVert_Source																					
MultiStateButton																					
PicturePushButton																					
PagingContainer																					
PictureToggleButton																					

3.0 WIDGET LIBRARY

Widgets \ Events	A661_EVT_CURSOR_ENTER	A661_EVT_CURSOR_INSIDE	A661_EVT_CURSOR_EXIT	A661_EVT_CURSOR_POS_CHANGE	A661_EVT_EDITBOX_OPENED	A661_EVT_FIRST_VIS_ENTRY_CHANGE	A661_EVT_FRAME_POS_CHANGE	A661_EVT_GESTURE_DIR_FLICK	A661_EVT_GESTURE_DOUBLE_TAP	A661_EVT_GESTURE_DRAG	A661_EVT_GESTURE_FLICK	A661_EVT_GESTURE_HOLD	A661_EVT_GESTURE_PINCH	A661_EVT_GESTURE_PRESS_AND_HOLD	A661_EVT_GESTURE_ROTATE	A661_EVT_GESTURE_TAP	A661_EVT_INCREMENT	A661_EVT_ITEM_IN_BOUNDS	A661_EVT_ITEM_OUT_OF_BOUNDS	A661_EVT_ITEM_SYNCHRONIZATION	A661_EVT_KEY
PopUpMenu																					
PopUpMenuButton																					
PopUpPanel																					
PopUpPanelButton																					
ProxyButton																					
PushButton																					
ScrollList						X															
ScrollPanel							X														
SelectionListButton																					
Slider																					
SymbolPushButton																					
SymbolToggleButton																					
TabbedPanelGroup																					
ToggleButton																					
TouchArea																					
WatchdogContainer																					

Table 3.1.4-2 – Widget Event Cross Reference

Widgets \ Events	A661_EVT_OFFSCREEN	A661_EVT_ONSCREEN	A661_EVT_POPUP_CLOSED	A661_EVT_POPUP_PANEL_CLOSED	A661_EVT_SEL_ENTRY_CHANGE	A661_EVT_SELECTION	A661_EVT_SELECTION_MAP	A661_EVT_STATE_CHANGE	A661_EVT_STRING_CHANGE	A661_EVT_STRING_CHANGE_ABORTED	A661_EVT_STRING_CONFIRMED	A661_EVT_TABBED_PANEL_CHANGE	A661_EVT_TOUCH	A661_EVT_VALUE_CHANGE	A661_EVT_VISIBLE_CHILD	A661_EVT_WATCHDOG_EXPIRED	A661_EVT_WATCHDOG_NORMAL
ActiveArea						X											
CheckButton								X									
ComboBox					X												
ComboBoxEdit					X				X	X	X						
CursorOver																	
CursorPosOverlay																	
EditBoxMasked									X	X	X						
EditBoxMultiLine									X	X	X						
EditBoxNumeric									X	X	X						
EditBoxNumericBCD									X	X	X						
EditBoxText									X	X	X						
GestureArea																	
KeyboardArea																	
ScrollWheelArea																	
MapHorz																	
MapHorz_Container	X	X															
MapHorz_ItemList						X											
MapHorz_Source							X										
MapVert																	
MapVert_ItemList						X											
MapVert_Source							X										
MultiStateButton								X									
PicturePushButton						X											
PagingContainer															X		
PictureToggleButton								X									
PopUpMenu			X														
PopUpMenuButton			X														
PopUpPanel				X													
PopUpPanelButton				X													
ProxyButton						X											
PushButton						X											
ScrollList					X												
ScrollPanel																	
Selector					**X**												
SelectionListButton					X												

Widgets \ Events	A661_EVT_OFFSCREEN	A661_EVT_ONSCREEN	A661_EVT_POPUP_CLOSED	A661_EVT_POPUP_PANEL_CLOSED	A661_EVT_SEL_ENTRY_CHANGE	A661_EVT_SELECTION	A661_EVT_SELECTION_MAP	A661_EVT_STATE_CHANGE	A661_EVT_STRING_CHANGE	A661_EVT_STRING_CHANGE_ABORTED	A661_EVT_STRING_CONFIRMED	A661_EVT_TABBED_PANEL_CHANGE	A661_EVT_TOUCH	A661_EVT_VALUE_CHANGE	A661_EVT_VISIBLE_CHILD	A661_EVT_WATCHDOG_EXPIRED	A661_EVT_WATCHDOG_NORMAL
Slider														X			
SymbolPushButton						X											
SymbolToggleButton								X									
TabbedPanelGroup												X					
ToggleButton								X									
TouchArea													X				
WatchdogContainer																X	X

3.2 HMI Widget Library Summary

3.2.1 Widget Summary

Table 3.2.1 summarizes the Widget Library.

Table 3.2.1 – Widget Library Summary

Widget Type	Description
ActiveArea	The Active Area widget is a transparent rectangular area defining an interactive area. Selection of this area will send an event to the owner application.
BasicContainer	The BasicContainer widget manages the visibility and the interactivity of a group of widgets.
BlinkingContainer	The purpose of the BlinkingContainer widget is to apply blinking behavior to a group of widgets.
BufferFormat	This widget provides a means for compressing data from different widgets (in the same layer) in one buffer. Use for this widget could be initialization of a page or refresh of big widget. The BufferFormat format is defined at definition phase. The content of the buffer is exchanged at run-time from the UA to the CDS.
CheckButton	A CheckButton allows the crew-member to select or deselect an option (other names: Radio/Toggle box).
ComboBox	A ComboBox is a widget providing a means to select one item among a list. This widget is composed of a static part displaying the selected item and a popup part displaying the list of selectable items.

Widget Type	Description
Connector	The purpose of this widget is to make a layer be the child of a container located in another layer. Typical use cases are for TabbedPanelGroup and MapHorz, which can mix data from several User Applications.
CursorPosOverlay	A CursorPosOverlay consists of a transparent rectangular area of the display. The distinguishing characteristic of a CursorPosOverlay is that the reportable event is the current cursor pointer position relative to the CursorPosOverlay position.
EditBoxMasked	The Masked edit box is an extension of the Text edit box. The difference with the basic Text edit box is that some characters are not modifiable by the crew member. Those Characters non-modifiable are specified by the user application by setting to 0 the “alpha mask” parameter and the “numeric mask”.
EditBoxNumeric	The numeric edit box allows editing a numeric value. A crew member can modify this value using its input devices. As it is a numeric value, CDS is able to increment itself the value. The widget can receive a number of increment or a numeric key value.
EditBoxText	A text edit box allows displays a string which can be modified by a crew member (other names: Text field, Text entry box).
GpArcEllipse	The graphical primitive GpArcEllipse allows the definition of an arc (portion of an ellipse or a circle).
GpArcCircle	The graphical primitive GpArcCircle allows the definition of a circular arc.
GpCrown	The graphical primitive GpCrown allows the definition of a circular filled region.
GpLine	The graphical primitive GpLine allows the definition of a line.
GpLinePolar	The graphical primitive GpLinePolar allows the definition of a line using polar coordinates.
GpRectangle	The graphical primitive GpRectangle allows the definition of a rectangle.
GpTriangle	The graphical primitive GpTriangle allows the definition of a triangle.
Label	A Label consists of a non-editable text field at a defined display location.
LabelComplex	A Complex Label consists of a non-editable text field at a defined display location. The graphical representation is managed through an escape sequence.
MapHorz_ItemList	MapHorz_ItemList represents a group of related graphics. Example use of the widget is the creation of flight plan, map background symbols, and TCAS intruders.

Widget Type	Description
MapHorz_Source	MapHorz_Source is a specialized container. It contains widgets expressed in a common coordinate system. It describes characteristics of the common coordinate system.
MapHorz	MapHorz consists of a rectangular region on the display, which contains reference information to allow the display of map features in the cockpit. It allows multiple sources of information with different coordinate systems to be fused into a composite map image.
MaskContainer	MaskContainer is intended to apply a referenced mask to a group of widgets.
Panel	A panel groups several widgets together in a rectangular area and has clipping capabilities.
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).
PicturePushButton	A momentary switched button with Picture, which allows the crew member to launch an action (to send an event to the owner user application).
PictureToggleButton	A button with two stable states and a Picture.
PopUpPanel	PopUpPanel is a container temporarily displayed on top of all other layers in its containing window. Because of this, PopUpPanel should not be used as a regular container.
PopUpMenu	PopUpMenu is a set of selectable items. This menu is displayed on top of all other layers in its containing window.
PopUpMenuButton	PopUpMenuButton is a button providing the ability to display a PopUpMenu.
PushButton	Momentary switched button, which allows the crew-member to launch an action (to send an event to the owner user application).
RadioBox	Manages the visibility and the interactivity of a group of CheckButtons or ToggleButtons. The selection of one of the CheckButtons or the ToggleButtons is exclusive.
RotationContainer	A RotationContainer has the same capabilities of a BasicContainer. It allows a rotation transformation to be applied to the descendants of the container.
ScrollPanel	A ScrollPanel is a “sheet container widget” of which only a subpart, the “frame,” is visible. Scroll controls provide the capability to scroll the visible part inside the whole sheet.
ScrollList	A ScrollList is a list of items of which only a subpart is visible. Scroll controls provide the capability to scroll the visible part inside the whole list.
Symbol	Symbol has rotation and color capability. Symbol widget has a reference to a table.

Widget Type	Description
TabbedPanel	A TabbedPanel widget is a Panel associated with a selection button. This widget is only for use within a TabbedPanelGroup widget.
TabbedPanelGroup	A TabbedPanelGroup groups several TabbedPanel widgets. A TabbedPanelGroup allows the user application or a crew member using a selection button to display one of the TabbedPanel widgets. All of the panels inside the TabbedPanel widgets occupy the same display space. Only one may be displayed at a time.
ToggleButton	A button with two stable states and some text.
TranslationContainer	A TranslationContainer is similar to a BasicContainer. It allows a translation transformation to be applied to the descendants of the TranslationContainer.
WIDGETS ADDED FOR SUPPLEMENT 1	
ComboBoxEdit	Like ComboBox, ComboBoxEdit provides a means to select one item in a list of items. This widget is composed of a static part displaying the selected item and a popup part displaying the list of selectable items. The static part of ComboBoxEdit contains a crew-modifiable text Label.
EditBoxMultiLine	EditBoxMultiLine is a text edit box for displaying text across several lines in a scrolling area.
ExternalSource	The function of the ExternalSource widget is to specify to the CDS where an external input should appear on the display. For example, an external input may be a video signal input or a bitmap image.
MapGrid	MapGrid provides a means for conveying arrays of data to the CDS that are rendered as area fills. The intended use is for filling areas on background layers of a Navigation window with colors and/or patterns that indicate terrain topography, precipitation intensity, or other irregular, dynamic data.
MapVert	The MapVert widget is the counterpart of the MapHorz widget for a vertical display made of a slice presentation. It is based on a Cartesian coordinate system.
MapVert_ItemList	The MapVert_ItemList is equivalent to the MapHorz_ItemList for vertical displays. A MapVert_ItemList contains a list of Items to be drawn.
MapVert_Source	The MapVert_Source is the equivalent of the MapHorz_Source for vertical displays. The MapVert_Source widget is a specialized container. It contains some MapVert_ItemList widgets to display. Items are expressed in a common coordinate system.

Widget Type	Description
MenuBar	A MenuBar is a widget containing button-type widgets (CheckButton, PushButton, PicturePushButton, etc.). It groups the buttons and implements specific behaviors to move from one button to another.
WIDGETS ADDED FOR SUPPLEMENT 2	
MutuallyExclusiveContainer	The MutuallyExclusiveContainer groups children widgets and provides control to assure that only one child is visible at the same time. The MutuallyExclusiveContainer has no graphical representation.
PictureAnimated	A PictureAnimated widget contains a reference to a set of images available in the CDS that cannot be modified by the user application. By displaying this set of pictures successively at a frequency defined as a parameter of the widget, the CDS performs an animation.
ProxyButton	The ProxyButton directs selection events from physical keys present in the CDS to a target widget.
SelectionListButton	The SelectionListButton allows a crew member to select one entry within a list. This widget is composed of a fixed label and a popup part displaying the list of selectable of items.
Slider	A Slider allows the crewmember to select a value between the range of MIN_VALUE and MAX_VALUE. The Slider can be displayed in either the horizontal or vertical axis.
SymbolAnimated	The SymbolAnimated widget is defined by a sequence of symbol references, along with orientations and relative movements for each cycle of the animation.
WatchdogContainer	This widget is used to assure certain sets of parameters are refreshed at a defined rate. If the timer is not refreshed at the required rate the CDS sends an event to the UA and automatically displays a predefined child widget.
WIDGETS ADDED FOR SUPPLEMENT 3	
CursorRef	This widget is used to define a screen or map location that can be used with the A661_REQ_CURSOR_ON_WIDGET command.
CursorOver	This widget is similar to the ActiveArea widget but generates events as soon as the cursor enters the widget's active area.
EditBoxNumericBCD	The EditBoxNumericBCD is very similar to EditBoxNumeric and has the same general features except that it allows the entry of non-base 10 values such as latitude or time.
FocusLink	This widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.

Widget Type	Description
FocusIn	Together with FocusOut, this widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.
FocusOut	Together with FocusIn, this widget is used to define a sequence of NextFocusedWidgets that crosses between one layer and another.
SizeToFitContainer	This widget is used to dynamically size a set of child widgets so that they are all the same size. The function can be applied in the vertical or horizontal axis.
ShuffleToFitContainer	This widget is used to arrange a set of child widgets so that there is no unused space between them. If a child is made invisible, all of the remaining children are moved to close up the space.
WIDGETS ADDED FOR SUPPLEMENT 4	
PopUpPanelButton	A PopUpPanelButton widget provides a way to select a container similar to a PopUpPanel. This widget is composed of a static part (a button) and a popup part. The popup part is similar to a PopUpPanel.
SymbolPushButton	A SymbolPushButton widget is a PushButton including a Symbol and possibly a text string.
SymbolToggleButton	A SymbolToggleButton widget is a button with two stable states, a Symbol and possibly a text string.
WIDGETS ADDED FOR SUPPLEMENT 5	
BroadcastReceiver	Provide a means for broadcasting one value to different widgets in one layer
DataScalingFR180	Provide a means for scaling one fr(180) value for one widget parameter.
DataScalingLong	The objective of this widget is to provide a means for scaling one long value for one widget parameter.
DataScalingULong	Provide a means for scaling one ulong value for one widget parameter.
GpPolyline	The GpPolyline widget is a graphical primitive that groups a large number of line segments into a single widget. This reduces processing for large sequences of connected lines.
MapHorz_Container	MapHorz_Container provides a way to position, rotate, and scale widgets on a map based upon real world position data and map range data. This is intended for use with non-interactive widgets.
MapHorz_Panel	MapHorz_Panel is a geographic aware container, used to present other widgets bound to a specific location in a map.

Widget Type	Description
NoServiceMonitor	The NoServiceMonitor is a non-graphical container widget used to display information when there is a communication problem between the UA and CDS.
NumericReadout	NumericReadout provides a way to display a dynamically changing numeric value and accompanying text legend.
PagingContainer	The PagingContainer widget is similar to the MutuallyExclusiveContainer and TabbedPanelGroup, but it allows the crew member to interactively choose which child container to display. It can be used as a means of driving animations or providing graphical feedback to the user.
WIDGETS ADDED FOR SUPPLEMENT 6	
AnimationGroup	This widget provides a means for grouping several animations in parallel or sequentially.
AnimationOnParam	This widget provides a means for performing an animation on one widget and a parameter.
AnimationRotation	This widget provides a means for performing a rotation animation on one widget.
AnimationScale	This widget provides a means for performing a scale animation on one widget.
AnimationTranslation	This widget provides a means for performing a translation animation on one widget.
DataConnector	**The DataConnector is similar to a regular Connector widget, but in addition, it provides the ability for an array of parametric information to be passed to the Application that owns the referenced layer.**
EventHandler	The EventHandler widget defines how CDS should handle an event without the UA needing to send Set Parameter commands after receiving the event. This widget forms a logical connection between an event and widget parameters for cases in which a UA responds to an event the same way each time.
FramingRefreshContainer	Utility container widget, which ensures that its children are rendered together, even though updates of its children are sent in separate set parameter commands. Supporting the display of a consistent data-context across widgets.
GestureArea	The GestureArea widget allows a UA to be notified of gesture events generated from user interaction over the widget area. With this widget, a UA can receive gesture data in discrete events with minimal interpretation.
InkArea	The InkArea allows a crew member to draw in an active area region of the widget using input from a digitizer or similar device. The impact to drawn content when the widget position or size is changed is implementation dependent.

Widget Type	Description
KeyboardArea	The KeyboardArea is a transparent rectangular widget. The objective of this widget is to send keyboard key press events from Crew member interactions. The principle is that this KeyboardArea widget defines a zone inside which the key press inputs are redirected to UA while the Cursor is over the area and when not consumed by another widget.
MapBoundary	MapBoundary defines a map boundary that can impact map items which are in descendent map widgets.
MapHorz_VertexBuffer	Contains vertex data to be used by vertex-array based rendering of graphical primitives. MapItems may reference vertex data contained within these widgets to render geometry. Vertices are referenced either sequentially or via a sequence of indices.
MapVert_Container	MapVert_Container provides a way to position and scale widgets on a map based upon real world position data and map range data. This is intended for use with non-interactive widgets.
MapVert_Panel	MapVert_Panel is a geographic aware container, used to present other widgets bound to a specific location in a map.
MultiStateButton	A MultiStateButton widget is a multiple stable-state button with text. It is similar to the ToggleButton widget but supports more than two states.
ScrollWheelArea	The ScrollWheelArea is transparent rectangular widget. The objective of this widget is to send Increment events from Crew member interactions (from scrollwheel interactions) in order to allow UA to simulate scrolling action at screen.
TouchArea	The TouchArea widget allows the UA to be notified of touch inputs over the widget. With this widget, a UA can receive and interpret touch data without being limited to the CDS supported gesture types and gesture interpretation.
WIDGETS ADDED FOR SUPPLEMENT 7	
GpTriangleFan	**The graphical primitive GpTriangleFan allows definition of a shape composed out of a fan of triangles.**
GpTriangleStrip	**The graphical primitive GpTriangleStrip widget enables the definition of a shape composed out of a linked strip of triangles.**
MapExternalSource	**The function of the MapExternalSource widget is to specify to the CDS where an external input should appear on the display, under a MapHorz or MapVert widget.**
ScaleContainer	**A ScaleContainer widget applies a scaling transformation to a group of widgets.**

Widget Type	Description
Selector	The Selector widget allows the user to select a single entry from a set of entries displayed according to the specified graphical layout.
Tree	A Tree widget is a set of items organized as a tree. A Tree widget behaves similarly to the ScrollList widget, except that the entries are grouped hierarchically.

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 widgets inside the HMI pages.
Graphical Representation	Category of widgets which have a graphical representation.
Interactive	Category of widgets with which crew members 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 widgets inside maps. Typical use case for this symbol is a Navigation Display format.
Text string	Category of widget that display a string of text.
UA Validation	Category of widgets which may have their events (if applicable) validated by the UA. The codes are defined as follows: A. Supports A661_ENABLE but UA Validation is not applicable for this widget type. **B. Supports A661_ENABLE with value A661_TRUE_WITH_VALIDATION and uses the A661_ENTRY_VALID parameter.** See Section 3.2.9 for more information.
Utility	Category of widgets that are not containers, do not have graphical representation, and are not interactive. These widgets have specific functionality in order to extend or optimize the ARINC 661 defined principles.

Table 3.2.2-2 – Widget Classification Table

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

	Widget Categories/Widgets	Container	Map Management	Graphical Representation	Text string	Interactive	Utility	UA Validation
3.3.42	TranslationContainer	X						
WIDGETS ADDED FOR SUPPLEMENT 1								
3.4.1	MapGrid		X	X				
3.4.2	ExternalSource							
3.4.3	MapVert	X	X					A
3.4.4	MapVert_Source	X	X			X		A
3.4.5	MapVert_ItemList		X	X	X	X		B
3.4.6	EditBoxMultiLine			X	X	X		B
3.4.7	ComboBoxEdit			X	X	X		B
3.4.8	MenuBar	X				X		A
WIDGETS ADDED FOR SUPPLEMENT 2								
3.5.1	MutuallyExclusiveContainer	X						A
3.5.2	ProxyButton					X		A
3.5.3	WatchdogContainer	X						
3.5.4	Slider			X		X		B
3.5.5	PictureAnimated			X				
3.5.6	SymbolAnimated			X				
3.5.7	SelectionListButton			X	X	X		B
WIDGETS ADDED FOR SUPPLEMENT 3								
3.6.1	EditBoxNumeric BCD			X	X	X		B
3.6.2	CursorRef						X	
3.6.3	CursorOver			X		X		A
3.6.4.1	FocusLink						X	
3.6.4.2	FocusIn						X	
3.6.4.3	FocusOut						X	
3.6.5	SizeToFitContainer	X		X				A
3.6.6	ShuffleToFitContainer	X		X				A
WIDGETS ADDED FOR SUPPLEMENT 4								
3.7.1	SymbolPushButton			X	X	X		B
3.7.2	SymbolToggleButton			X	X	X		B
3.7.3	PopUpPanelButton	X		X	X	X		
WIDGETS ADDED FOR SUPPLEMENT 5								
3.8.1	GpPolyline			X				
3.8.2	PagingContainer	X		X		X		B
3.8.3	NumericReadout			X	X			
3.8.4	MapHorz_Container	X	X					
3.8.5	MapHorz_Panel	X	X	X				A
3.8.6	DataScalingLong						X	
3.8.7	DataScalingULong						X	
3.8.8	DataScalingFR180						X	
3.8.9	BroadcastReceiver						X	
3.8.10	NoServiceMonitor	X					X	
WIDGETS ADDED FOR SUPPLEMENT 6								
3.9.1	MultiStateButton			X	X	X		B
3.9.2	KeyboardArea					X		**A**
3.9.3	ScrollWheelArea					X		**A**

	Widget Categories/Widgets	Container	Map Management	Graphical Representation	Text string	Interactive	Utility	UA Validation
3.9.4	MapHorz_VertexBuffer		X					
3.9.5	TouchArea			X		X		A
3.9.6	GestureArea			X		X		A
3.9.7	InkArea			X		X		
3.9.8	AnimationTranslation						X	
3.9.9	AnimationScale						X	
3.9.10	AnimationGroup	X					X	
3.9.11	AnimationRotation						X	
3.9.12	AnimationOnParam						X	
3.9.13	MapVert_Container	X	X					
3.9.14	MapVert_Panel	X	X	X				A
3.9.15	MapBoundary	X	X					A
3.9.16	DataConnector						X	A
3.9.17	EventHandler						X	
3.9.18	FramingRefreshContainer	X					X	
WIDGETS ADDED FOR SUPPLEMENT 7								
3.10.1	**ScaleContainer**	**X**						
3.10.2	**Selector**			**X**	**X**	**X**		**B**
3.10.3	**Tree**			**X**		**X**		**B**
3.10.4	**MapExternalSource**							
3.10.5	**GpTriangleFan**			**X**				
3.10.6	**GpTriangleStrip**			**X**				

3.2.3 Container

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

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

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

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

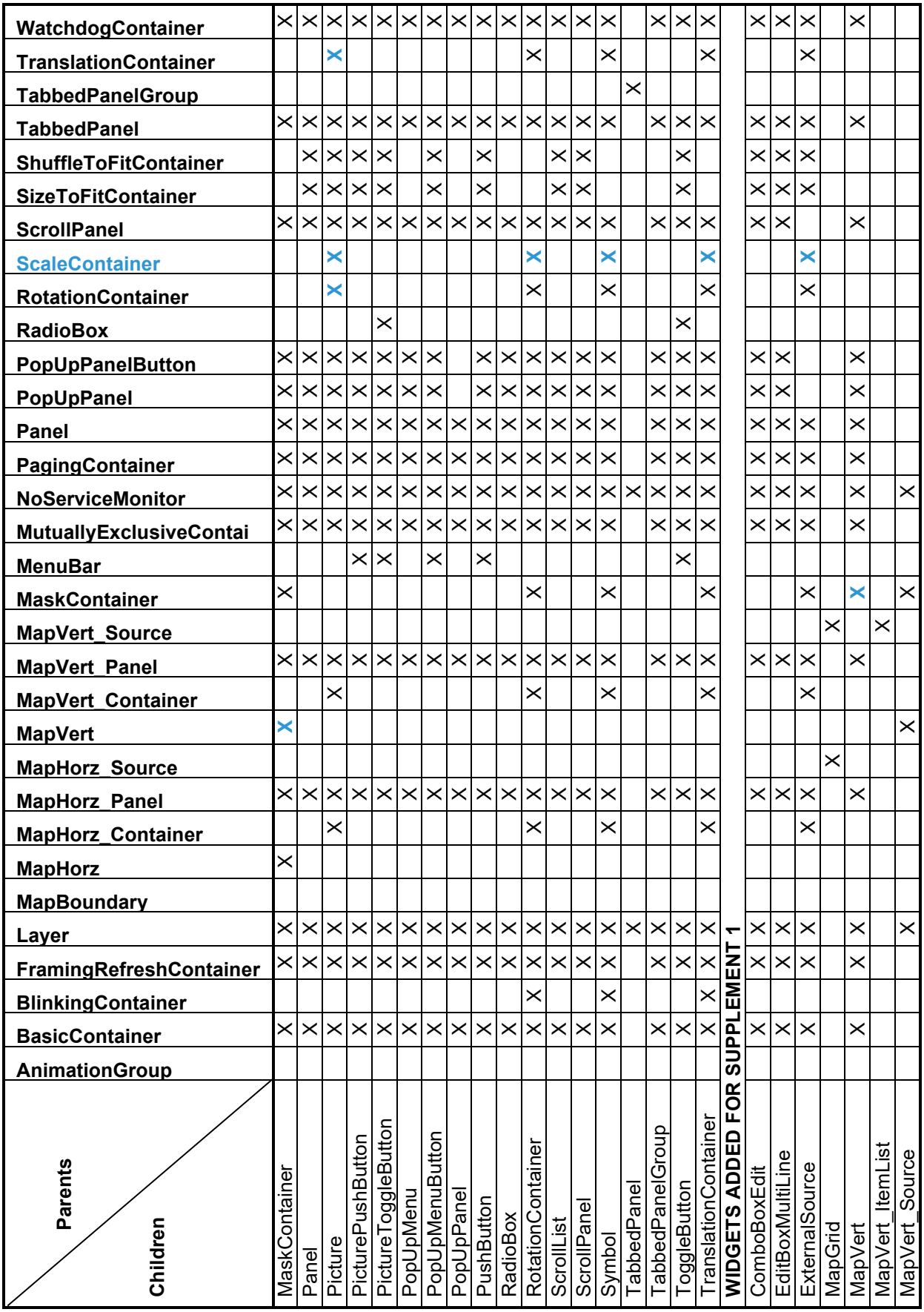
Table 3.2.3.1 describes the possible children of Container widgets. Although an ARINC 661 Layer is not a widget, it has been listed with the Containers because of its capability to be the parent of widgets.

3.2.3.1 Possible Children of Container Widgets

Possible Children of Container Widgets is defined in Tables 3.2.3.1.

Table 3.2.3.1 – Possible Children of Container Widgets

Parents \ Children	ActiveArea	BasicContainer	BlinkingContainer	BufferFormat	CheckButton	ComboBox	Connector	CursorPosOverlay	EditBoxMasked	EditBoxNumeric	EditBoxText	GpArcCircle	GpArcEllipse	GpCrown	GpLine	GpLinePolar	GpRectangle	GpTriangle	Label	LabelComplex	MapHorz	MapHorz_ItemList	MapHorz_Source	MapLegacy (DELETED)
WatchdogContainer	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
TranslationContainer			X				X					X	X	X	X	X	X	X	X					
TabbedPanelGroup							X																	
TabbedPanel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
ShuffleToFitContainer	X				X	X	X		X	X	X						X		X	X				
SizeToFitContainer	X				X	X	X		X	X	X						X		X	X				
ScrollPanel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
ScaleContainer			X				X					X	X	X	X	X	X	X	X					
RotationContainer			X				X					X	X	X	X	X	X	X	X					
RadioBox					X		X																	
PopUpPanelButton	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
PopUpPanel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
Panel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
PagingContainer	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
NoServiceMonitor	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	
MutuallyExclusiveContai	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
MenuBar					X		X																	
MaskContainer			X				X					X	X	X	X	X	X	X	X		X		X	
MapVert_Source																								
MapVert_Panel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
MapVert_Container			X				X					X	X	X	X	X	X	X	X					
MapVert							X																	
MapHorz_Source																						X		
MapHorz_Panel	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
MapHorz_Container			X				X					X	X	X	X	X	X	X	X					
MapHorz							X																X	
MapBoundary							X																X	
Layer	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	
FramingRefreshContainer	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
BlinkingContainer							X					X	X	X	X	X	X	X						
BasicContainer	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
AnimationGroup							X																	


Parents
Children
WatchdogContainer
TranslationContainer
TabbedPanelGroup
TabbedPanel
ShuffleToFitContainer
SizeToFitContainer
ScrollPanel
ScaleContainer
RotationContainer
RadioBox
PopUpPanelButton
PopUpPanel
Panel
PagingContainer
NoServiceMonitor
MutuallyExclusiveContai
MenuBar
MaskContainer
MapVert_Source
MapVert_Panel
MapVert_Container
MapVert
MapHorz_Source
MapHorz_Panel
MapHorz_Container
MapHorz
MapBoundary
Layer
FramingRefreshContainer
BlinkingContainer
BasicContainer
AnimationGroup
MaskContainer
Panel
Picture
PicturePushButton
PictureToggleButton
PopUpMenu
PopUpMenuButton
PopUpPanel
PushButton
RadioBox
RotationContainer
ScrollList
ScrollPanel
Symbol
TabbedPanel
TabbedPanelGroup
ToggleButton
TranslationContainer
WIDGETS ADDED FOR SUPPLEMENT 1
ComboBoxEdit
EditBoxMultiLine
ExternalSource
MapGrid
MapVert
MapVert_ItemList
MapVert_Source

Children \ Parents	MenuBar	WIDGETS ADDED FOR SUPPLEMENT 2	MutuallyExclusiveContainer	PictureAnimated	ProxyButton	SelectionListButton	Slider	SymbolAnimated	WatchdogContainer	WIDGETS ADDED FOR SUPPLEMENT 3	CursorOver	CursorRef	EditBoxNumericBCD	FocusIn	FocusLink	FocusOut	ShuffleToFitContainer	SizeToFitContainer	WIDGETS ADDED FOR SUPPLEMENT 4	PopUpPanelButton	SymbolPushButton	SymbolToggleButton	WIDGETS ADDED FOR SUPPLEMENT 5	BroadcastReceiver	DataScalingFR180
AnimationGroup																									
BasicContainer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
BlinkingContainer																									
FramingRefreshContainer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
Layer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X		X	X
MapBoundary																									
MapHorz																									
MapHorz_Container				X				X				X													
MapHorz_Panel	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
MapHorz_Source																									
MapVert																									
MapVert_Container				X				X				X													
MapVert_Panel	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
MapVert_Source																									
MaskContainer								X																	
MenuBar					X															X		X			
MutuallyExclusiveContainer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
NoServiceMonitor	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
PagingContainer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
Panel	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
PopUpPanel	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
PopUpPanelButton	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
RadioBox																						X			
RotationContainer				X				X																	
ScaleContainer				X				X																	
ScrollPanel			X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			
SizeToFitContainer				X		X	X				X	X	X				X	X		X	X	X			
ShuffleToFitContainer				X		X	X				X	X	X				X	X		X	X	X			
Slider																									
SelectionListButton																									
ProxyButton																									
PictureAnimated																									
MutuallyExclusiveContainer																									
MenuBar																									
WatchdogContainer																									
SymbolAnimated																									
SymbolPushButton																									
SymbolToggleButton																									
TabbedPanel	X		X	X	X	X	X	X	X		X		X	X	X	X	X	X		X	X	X			
TabbedPanelGroup																									
TranslationContainer				X				X																	
WatchdogContainer	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X			

WIDGETS ADDED FOR SUPPLEMENT 2

WIDGETS ADDED FOR SUPPLEMENT 3

WIDGETS ADDED FOR SUPPLEMENT 4

WIDGETS ADDED FOR SUPPLEMENT 5

Children \ Parents	DataScalingLong	DataScalingULong	GpPolyline	MapHorz_Container	MapHorz_Panel	NoServiceMonitor	NumericReadout	PagingContainer	**WIDGETS ADDED FOR SUPPLEMENT 6**	AnimationTranslation	AnimationScale	AnimationGroup	AnimationRotation	AnimationOnParam	DataConnector	EventHandler	FramingRefresh Container	GestureArea	InkArea	KeyboardArea	MapHorz_VertexBuffer	MapVert_Container	MapVert_Panel	MapBoundary	MultiStateButton
AnimationGroup										X	X	X	X	X	X										
BasicContainer			X				X	X							X		X	X	X	X					X
BlinkingContainer			X				X								X										
FramingRefreshContainer			X				X	X							X			X	X	X					X
Layer	X	X	X			X	X	X		X	X	X	X	X	X	X	X	X	X	X					X
MapBoundary															X									X	
MapHorz															X										
MapHorz_Container			X				X								X										
MapHorz_Panel			X				X	X							X		X	X	X	X					X
MapHorz_Source				X	X																X				
MapVert															X										
MapVert_Container			X				X								X										
MapVert_Panel			X				X	X							X		X	X	X	X					X
MapVert_Source																						X	X		
MaskContainer			X				X								X				X						
MenuBar															X										X
MutuallyExclusiveContai			X				X	X							X		X	X	X	X					X
NoServiceMonitor			X				X	X							X		X	X	X	X					X
PagingContainer			X				X	X							X		X	X	X	X					X
Panel			X				X	X							X		X	X	X	X					X
PopUpPanel			X				X	X							X		X	X	X	X					X
PopUpPanelButton			X				X	X							X		X	X	X	X					X
RadioBox															X										
RotationContainer			X				X								X										
ScaleContainer			X				X								X										
ScrollPanel			X				X	X							X		X	X	X	X					X
ShuffleToFitContainer															X			X	X	X					X
SizeToFitContainer															X			X	X	X					X
TabbedPanel			X				X	X							X		X	X	X	X					X
TabbedPanelGroup															X										
TranslationContainer			X				X								X										
WatchdogContainer			X				X	X							X		X	X	X	X					X

Parents / Children	ScrollWheelArea	TouchArea	WIDGETS ADDED FOR SUPPLEMENT 7	GpTriangleFan	GpTriangleStrip	MapExternalSource	ScaleContainer	Selector	Tree
WatchdogContainer	X	X		X	X		X	X	X
TranslationContainer				X	X		X		
TabbedPanelGroup									
TabbedPanel	X	X		X	X		X	X	X
ShuffleToFitContainer	X	X						X	X
SizeToFitContainer	X	X						X	X
ScrollPanel	X	X		X	X		X	X	X
ScaleContainer				X	X		X		
RotationContainer				X	X		X		
RadioBox									
PopUpPanelButton	X	X		X	X		X	X	X
PopUpPanel	X	X		X	X		X	X	X
Panel	X	X		X	X		X	X	X
PagingContainer	X	X		X	X		X	X	X
NoServiceMonitor	X	X		X	X		X	X	X
MutuallyExclusiveContai	X	X		X	X		X	X	X
MenuBar									
MaskContainer				X	X		X		
MapVert_Source									
MapVert_Panel	X	X		X	X		X	X	X
MapVert_Container				X	X		X		
MapVert						X			
MapHorz_Source									
MapHorz_Panel	X	X		X	X		X	X	X
MapHorz_Container				X	X		X		
MapHorz						X			
MapBoundary									
Layer	X	X		X	X		X	X	X
FramingRefreshContainer	X	X		X	X		X	X	X
BlinkingContainer				X	X		X		
BasicContainer	X	X		X	X		X	X	X
AnimationGroup									

3.2.4 Graphical Representation

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

3.2.5 Text Strings

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

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

- LabelComplex
- ScrollList

The escape capabilities are also available for the following MIL ItemList Items:

- LEGEND
- LEGEND_POP_UP

The escape sequences may be embedded in a text string to allow special formatting to occur. For widgets, the default graphic properties for the text are defined through the “DefaultStyleText” parameter. DefaultStyleText can also indicate whether or not escape sequences are in use. For MIL ItemList Items, the ITEM_STYLE item can indicate whether or not escape sequences are in use.

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. **If the character encoding is A661_ASCII_EXTENDED, each character has a single-byte representation. If the character encoding is A661_GBK or A661_UTF8, some characters have a multi-byte representation.**

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

3.2.5.1 Available Character Set and Character Set Encodings

This section defines characters available for all text strings defined in this specification. The characters in this table are representative of the shapes but are not intended to define a font.

Each Definition File has an associated character set encoding, which applies to all string type parameters in the Definition File, as well as the Run-time Parameters for widgets in the Definition File. There are three character set encodings available as part of this standard, with the possibility of adding custom ones. Refer to Table 4.5.3.2-6 and Section 6.2.1 for the way the character encoding is specified in the binary and XML DF, respectively.

- **ASCII Extended** character set encoding. Each character is encoded into one byte, based on the ASCII standard with A661-specific and possibly implementation-dependent characters added.
- **UTF-8** character set encoding, as defined in the Unicode standard (*http://www.unicode.org*).
- **GBK** character set encoding (Chinese Internal Code Specification).

All string length parameters are to be considered as a number of bytes (not a number of characters).If a custom character encoding is used, an encoding based on 8 bit code units (such as ASCII Extended, GBK, UTF-8) would tend to have better compatibility with legacy CDS and UA implementation code (especially if the encoding is compatible with ASCII) than an encoding based on 16 or 32 bit code units such as UTF-16 or UTF-32.For the standard characters found in Table 3.2.5.1-1, the same standard character code values are used for the binary encoding in ASCII Extended, UTF-8, and GBK character encoding cases since UNICODE and GBK share the same code point values as ASCII for the 0…127 range.

Table 3.2.5.1-1 – Available Character Set

Control Characters (ASCII/UNICODE/GBK)

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

Printing Characters (ASCII/UNICODE/GBK)

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

For characters outside the range of described in Table 3.2.5.1-1:

- **ASCII Extended encoding**: Table 3.2.5.1-2 below is provided to define standard ARINC 661 characters in that encoding. Implementation dependent characters are also permitted.
- **UTF-8 encoding**: characters are chosen from the UNICODE standard, and are not restricted to the ones from Table 3.2.5.1-2. Table 3.2.5.1-2 is only useful in that it documents the UNICODE code points of those characters. Implementation dependent characters not found in the UNICODE character set can use private use area code points reserved in UNICODE.
- **GBK encoding:** characters are chosen from the GBK standard and are not restricted to the ones from Table 3.2.5.1-2. Table 3.2.5.1-2 is only useful in that it documents the GBK code points of those characters. The handling of Implementation dependent characters not found in the GBK character set is implementation dependent. GBK has application-specific code points available, but Section 6 does not provide a mechanism to save such characters to the (UTF-8 encoded) XML DF format.

  Note: The Greek letters in the character set are a mixture of upper and lower case. This mixture is specific and intentional. Only those Greek characters considered useful in ARINC 661 applications are included.

Table 3.2.5.1-2 – A661 Character Set Extensions

ASCII Extended	Char	Description	Equivalent UNICODE Code Point	Equivalent GBK Code Point
h80	△	overfly triangle	h25B3	hA1F7
h81	°	degrees	hB0	hA1E3
h82	◇	diamond	h25C7	hA1F3
h83	▭	box	h25AD	*not available:* closest match is hA1F5
h84	←	Left arrow	h2190	hA1FB
h85	→	Right arrow	h2192	hA1FA
h86	↑	Up arrow	h2191	hA1FC
h87	↓	Down arrow	h2193	hA1FD
h88	α	Alpha	h03B1	hA6C1
h89	β	Beta	h03B2	hA6C2
h8A	γ	Gamma	h03B3	hA6C3
h8B	φ	Phi	h03C6	hA6D5
h8C	θ	Theta	h03B8	hA6C8
h8D	ψ	Psi	h03C8	hA6D7
h8E	ρ	Rho	h03C1	hA6D1
h8F	ε	Epsilon	h03B5	hA6C5
h90	π	Pi	h03C0	hA6D0
h91	ω	Omega	h03C9	hA6D8
h92	Δ	Delta	h0394	hA6A4
h93	τ	Tau	h03C4	hA6D3
h94	⬚	Figure Space	h2007	*not available:* closest match is hA1A1
h95	▲	**Filled up pointing triangle**	**h25B2**	**hA1F8**
h96	▼	**Filled down pointing triangle**	**h25BC**	**hA88B**
h97	►	**Filled right pointing triangle**	**h25B6**	***not available***
h98	◄	**Filled left pointing triangle**	**h25C0**	***not available***
h99	…	**Horizontal ellipsis**	**h2026**	**hA1AD**

In the ASCII Extended encoding case, the character set tables above define an ASCII Extended character mapping between ASCII Extended code point values and UNICODE code point values to allow conversions in both directions. Such conversions are useful:

- For XML to binary, or binary to XML conversions, since the XML file contents is based on UNICODE code points encoded using UTF-8.
- Possibly for implementation dependent purposes.

Each CDS implementation has an associated ASCII Extended character mapping, which can be the same as the mapping described in this section, or have adjustments (e.g., to add implementation dependent characters). Appendix L

defines an XML format which formalizes the definition of the ASCII Extended character mapping used for a particular implementation.

Each XML DF optionally has an associated ASCII Extended character mapping reference (see Section 6.2.1.1).

3.2.5.2 Notation Examples

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

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

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

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

“⊗” is the concatenation symbol.

Examples of concatenation follow:

Concatenation of strings -

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

Concatenation of sets -

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

3.2.5.3 Change Style Capabilities

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

Table 3.2.5.3 – Escape Sequence Types

Escape Capabilities	Escape Sequence Type	Description
Foreground Color	TForeColor	Sets the text color. Setting this Escape sequence in the middle of the string will cause all following text to be the new color.
Background Color	TBackColor	Sets the background fill color. Setting this Escape sequence in the middle of the string will cause all following text to have the new background color.
Font	TFont	Sets the font of the text. Setting this Escape sequence in the middle of the string will cause all following text to use the new font.
VideoInv	TVideoInv	Inverse video between the current foreground and background color. The inverse video is applied to characters between these two escape sequences: a Start and an End sequence.
Animation	TAnimation	Animation of text is applied to characters between two escape sequences: Start and End sequence.
Underline	TUnderline	Underline characters capability. Underlining is applied to characters between these two escape sequences: a Start and an End sequence.
Outline	TOutline	An Outline capability. It is the border definition around text.
Bold	TBold	Bold characters capability. It is applied to characters between these two escape sequences: a Start and an End sequence.
Crossed	TCrossed	~~Crossed characters~~ capability. It is applied to characters between these two escape sequences: a Start and an End sequence.
Framed	TFramed	Framed characters capability. It is applied to characters between these two escape sequences: a Start and an End sequence.
Occurrence	TOccur	Number of occurrences of a set of characters. It is applied to characters between these two escape sequences: a Start and an End sequence. The occurrence value number is the first hex value after the start sequence. It is a hex integer value from 0 to 255 representing the number of times the set of characters should appear.

3.2.5.4 Default Graphic Properties

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

3.2.5.5 Escape Sequences Description

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

Escape Sequences Descriptions are defined in Table 3.2.5.5-1.

Table 3.2.5.5-1 – Escape Sequences Description

Type	Starting Sequence	Ending Sequence	Escape Sequence
TOutline	-	-	ESC⊗Koutline⊗Tvalue0
TBackColor	-	-	ESC⊗KbackColor⊗Tvalue1
TForeColor	-	-	ESC⊗KforeColor⊗Tvalue1
TFont	-	-	ESC⊗Kfont⊗Tvalue2
TVideoInv	ESC⊗KvideoInv_B	ESC⊗KvideoInv_E	
TAnimation	ESC⊗Kanimation_B	ESC⊗Kanimation_E	
TUnderline	ESC⊗Kunderline_B	ESC⊗Kunderline_E	
TBold	ESC⊗Kbold_B	ESC⊗Kbold_E	
TCrossed	ESC⊗Kcrossed_B	ESC⊗Kcrossed_E	
TFramed	ESC⊗Kframed_B	ESC⊗Kframed_E	
TOccur	ESC⊗Koccur_B⊗P1	ESC⊗Koccur_E	

Where:

Tvalue0: Standard for Outline

Bit 1 = T (line on Top)

Bit 2 = B (line on Bottom)

Bit 3 = L (line on Left)

Bit 4 = R (line on Right)

Table 3.2.5.5-1a – Binary Value and Hex Code Description

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

Tvalue1 = {implementation dependent list}

Tvalue2 = {implementation dependent list}

Escape Identifiers are defined in Table 3.2.5.5-2.

Table 3.2.5.5-2 – Escape Identifier

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

P1 is a hex integer value from 0 to 255 representing the number of times the set of characters embedded between the escape sequences should appear. Thus, if P1 is set to one (1) the set of characters should appear one time, if P1 is set to two (2) the set of characters should appear two times, and so on. The meaning of a P1 value of zero (0) is implementation dependent.

The escape sequence should be treated as a sequence of bytes, where P1, Tvalue1, and Tvalue2 values are always stored in one byte, independent of the character encoding of the string. This requires special consideration for escape sequences in character encoding cases in which the escape sequence may not be consistent with the character encoding of the string:

- For example: P1, Tvalue1, or Tvalue2 values between 128 and 255 are not consistent with GBK or UTF-8 encodings of a string.
- See Section 6.2.4.2 for XML considerations of ARINC 661 escape sequences.
- For escape sequences that are not in strings, like in the DefaultStyleText parameter which is of type uchar*12, the charset encoding function is not used, and therefore the Tvalue1, Tvalue2 and P1 values are always encoded on one byte.

  Note: Designers wishing to use zero values for P1, Tvalue1, and Tvalue2 should be aware that it may make string parsing more difficult because of possible ambiguity between the numeric value zero and the NULL character, which is commonly used as a string terminator in this standard.

3.2.5.6 Text Alignment

Text is typically aligned relative to a rectangular region, e.g., as defined by an Alignment value, PosX, PosY, SizeX, and SizeY.

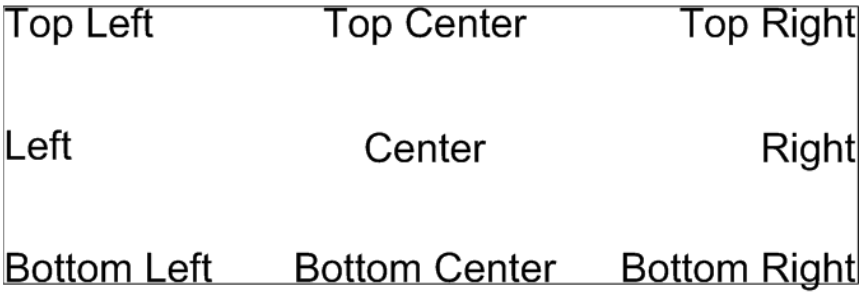


Figure 3.2.5.6-1 – Typical Text Alignment

Text can also be aligned according to a PosX, PosY position instead of a region (e.g., TEXT Symbol Graphical Definition Command in Section 5.2.3.12). In such a case, the PosX, PosY defines an anchor point and Alignment refers to the position of the anchor point with respect to the rendered text. For example, left-alignment anchors the text on the left hand side of the bounding box of the text; right-alignment anchors the text on the right side of the bounding box of the text. In the illustration below, the red cross marks the (PosX, PosY) anchor position (and would not be rendered as a cross). Multiline text could be handled by aligning the box of the multiple lines in a similar way.

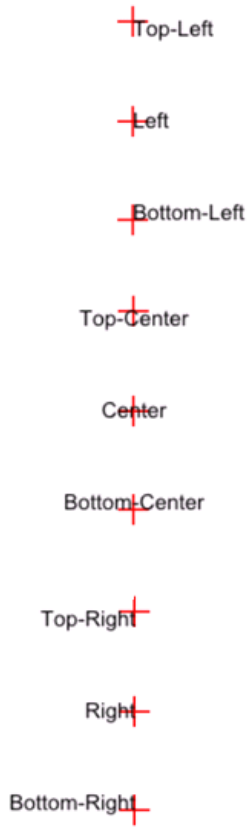


Figure 3.2.5.6-2 – Anchor Point Alignment

COMMENTARY

ARINC Specification 661 does not specify how text alignment should be performed as a function of font characteristics (baseline, ascent, descent, etc.), so there might be some small differences in placement between CDS implementations, both for alignment to a rectangular region and alignment to a point position. For example, the diagram above is not intended to be a requirement for the exact placement (at a pixel level), but just to give the general concept of where the text should be placed relative to the anchor position.

3.2.6 Formatted Numeric Values

Some widgets and MapItems, such as the EditBoxNumeric and NumericReadout widgets use a numeric value according to a format string.

The formatting of the numeric value depends on the pattern used for the format string. **The string is composed of any authorized characters. Some of them will be interpreted to format the value.**

The format string is on the form:

"[*](+)[_|~|*][#|*](.[#|*][_|~|*])"

where:

- **[A] means a set of 0 or more of A.**
- **(A) means a set of 0 or 1 of A.**
- **A | B means A or B.**
- **'_' indicates display of the digit extracted from the value if the digit is non-zero. This formatting element can be used to avoid display of leading and trailing zeroes.**
- **'~' indicates display of the digit extracted from the value if the digit is non-zero, otherwise a figure space is displayed (see Section 3.2.5.1). This formatting element can be used to ensure vertical alignment of numbers that do not have the same number of non-zero digits without explicitly displaying the leading and trailing zero digits.**
- **'#' indicates a display of the digit extracted from the value or of the character "0".**
- **'.' indicates the separator between integer part and decimal part of the value. Must be unique inside FormatString.**
- **'+' forces the sign character to be present in the final string at the defined position. Note that if this sign character is not present in the FormatString, the final string will be one character longer than the FormatString when the value is negative. In that case, the sign will be added at the beginning of the displayed string.**
- **'*' Any other characters (any except one of '+', '_', "~", '#', '.') will be interpreted as a mask character.**

  Note: '_' or "~" are not allowed between '#' and '.'.

Whether to show "+", "-", or nothing at all in front of the value 0 is implementation dependent.

Examples:

FormatString	Value	Displayed string
+____.__	-123.40	-123.4
+____.##	123.40	+123.40
___#.##	-23.45	-23.45
~~#.##	-23.45	- 23.45
_.~W	0	. W
_._W	0	.W
###°##′##″	402318	040°23′18″

3.2.7 Interactive

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

3.2.8 Dynamic Motion (Deprecated)

3.2.9 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: Horizontal and Vertical. In both cases interaction between widgets composing a Map is similar. See Table 3.2.2-2 for widgets classified as part for the Map Management group.

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

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

3.2.9.1 Horizontal Map Management

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

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

COMMENTARY

MapHorz_ItemList could be used to represent flight plan or map background symbols from an FM Application or TCAS intruders from a TCAS application.

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

The MapGrid widget draws map background data as a series of rectangles. Details about MapGrid can be found in Section 3.4.1.

Any Item in a MapHorz_ItemList has its position expressed in a local coordinate system, as opposed to the display unit or screen coordinate systems. 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 the local coordinate system type is contained in a MapHorz_Source widget. Data in MapHorz_ItemList is meaningless without the MapHorz_Source's MapDataFormat parameter.

Similarly, for MapGrid widgets, the IncrementX and IncrementY parameters are expressed in the real-world units defined in MapHorz_Source. As a result, each MapGrid and MapHorz_ItemList widget has to be a child of a MapHorz_Source.

The second step is to convert the real-world coordinate system values into the screen coordinate system. The MapHorz widget supports this conversion. The rationale behind splitting the transformation between MapHorz and MapHorz_Source was to allow objects from multiple world coordinate systems to be merged into one Map Image. Several MapHorz_ItemList and MapGrids can be merged into one Map, even if they use different world coordinate systems.

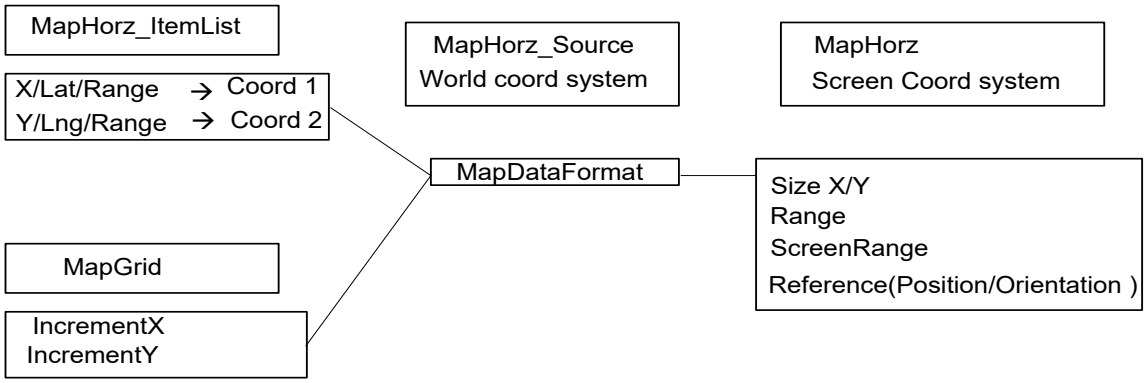


Figure 3.2.9.1 – Coordinate System for MapHorz Widget Management

A 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 of its MapHorz_ItemList's coordinate system through the MapHorz_Source widget.

A special UA is responsible for passing to the CDS all reference information required for the CDS to perform the merger. This UA, called the Master Application, federates all MapHorz_Source data to enable CDS to perform the merger. The Master Application provides reference information to the CDS through the MapHorz widget.

COMMENTARY

In the case of an ND window, one possible implementation would be for the ND application to be the Master Application while the FMS User Application, TCAS User Application and other UAs serve as Map Data User Applications.

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

3.2.9.1.1 Link Between MapHorz, MapHorz_Source, MapHorz_ItemList, and MapGrid

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

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

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

The MapHorz_Source defines coordinate system characteristics for all its contained descendants.

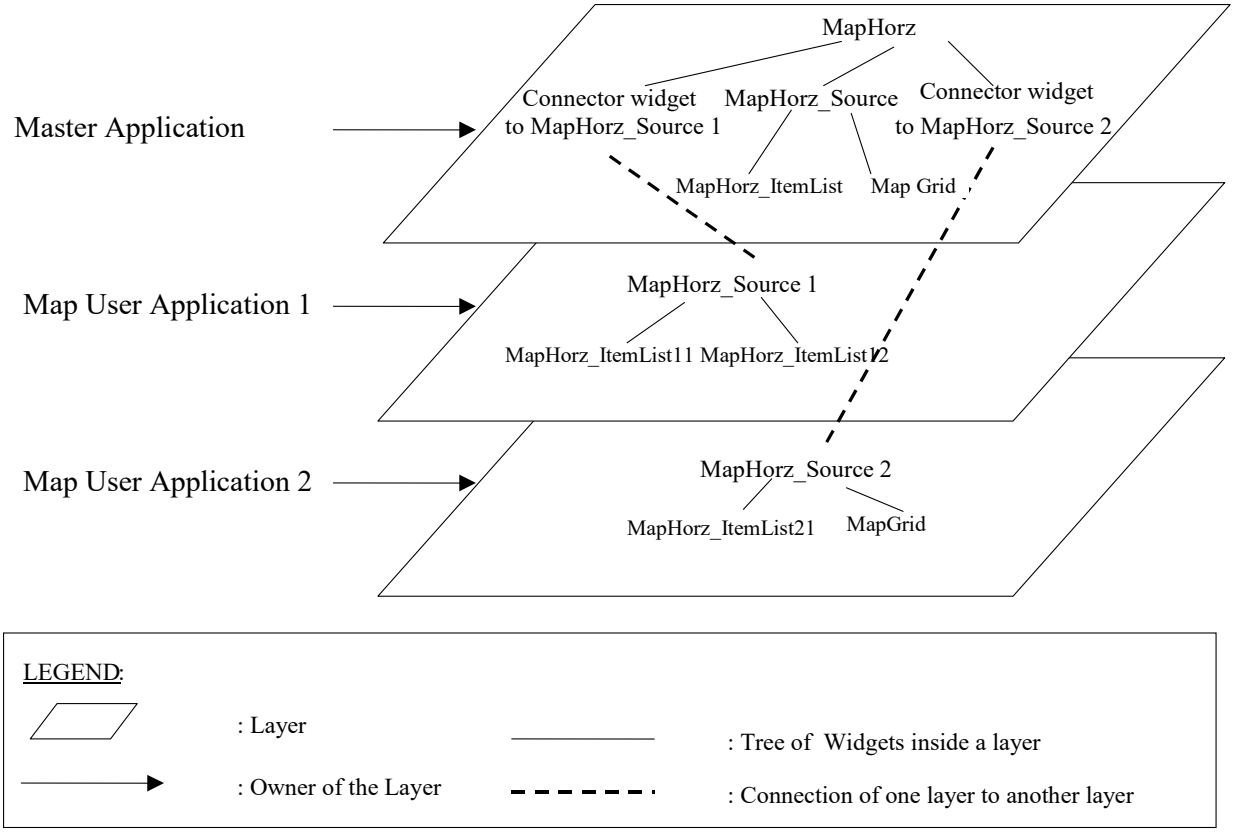


Figure 3.2.9.1.1 – Hierarchical Structure For MapHorz Widget Management

3.2.9.1.2 Parameter Definition for MapHorz and MapHorz_Source

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

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

The MAPHORZ widget is defined by the following parameters:

Table 3.2.9.1.2-1 – MAPHORZ Widget Parameters

MAPHORZ:	
- MAPHORZ X, Y:	Position of the MapHorz widget.
- PRP lat/lng:	PRP latitude and longitude.
- PRP X, Y :	PRP position on the display. Value relative to MAPHORZ X,Y.
- True North Angle :	Angle between True North and the Up direction of the display.
- Range:	Geo-referenced range expressed Range in nm.
- Screen Range :	Distance in screen units (hundredth of mm) equivalent to range
MAPHORZ_SOURCE:	
- Coordinate System :	Enumerated value. Lat/lng is one of these values.

3.2.9.2 Vertical Map Management

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

MapHorz→ MapVert

- MapHorz_Source→ MapVert_Source
- MapHorz_ItemList→ MapVert_ItemList
- MapGrid→ MapGrid

3.2.9.3 Priority Management

The drawing priority between widgets is defined as follows:

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

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

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

COMMENTARY

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

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

3.2.9.4 Map Synchronization Number

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

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

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

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

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

1. The Map Master internally decides to update range, scale, rotation, position, etc. of a map and does all necessary related computations and preparations.
2. The Map Master increments the Map Synchronization Number in the MapHorz widget for the affected window (this suppresses rendering of any incompatible MapHorz_ItemList) and then sends any updated control parameters to the MapHorz as applicable.
3. Then Map Master notifies the UAs of updated control parameters as applicable, through unspecified means (e.g., ARINC 664 Ethernet broadcast parameters).
4. The UA detects the changes in window control parameters and does all necessary related computations and preparations. There is no hurry, because the incompatible MapHorz_ItemLists are no longer being rendered due to the mismatch of the synchronization numbers.
5. UA sends an updated map item list to CDS (using 661 run-time messages) and then increments Map Synchronization Number for the updated MapHorz_ItemList widgets to match (thus allowing the widget to be drawn). The UA repeats update/increment sequence for other widgets or widget groups if multiple update cycles are needed for full refresh.

3.2.9.5 MapItem Legends

The following MapItems (called LEGENDxxx in the following paragraph) use Legend Strings to present Legends or format numeric values in MapHorz_ItemList or MapVert_ItemList widgets:

- LEGEND
- LEGEND_COMBO
- LEGEND_HIGHLIGHT
- LEGEND_POP_UP
- LEGEND_READOUT_FORMAT_STRING. Note that for this MapItem, the LEGEND_READOUT MapItem present the Legend, by using both the numeric value provided by the LEGEND_READOUT MapItem, and the format provided by the LEGEND_READOUT_FORMAT_STRING MapItem

  Notes:

  1. A LEGEND, LEGEND_COMBO, LEGEND_HIGHLIGHT, LEGEND_POP_UP, or LEGEND_READOUT Item can follow a LEGEND_ANCHOR Item in a Map Item List. Any of these items can also follow a SYMBOL Item, indicating that the Legend is to be displayed with the Symbol. Positioning/alignment of Legend text relative to the LEGEND_ANCHOR or SYMBOL X,Y position can be controlled by the UA through the ITEM_STYLE Item.
  2. LEGEND, LEGEND_COMBO, LEGEND_HIGHLIGHT, LEGEND_POP_UP, or LEGEND_READOUT Item string appearance can be changed either by inserting ITEM_STYLE items between consecutive items, or by the use of Escape Sequences in the associated LEGENDxxx Item strings. The ITEM_STYLE item can also be used to indicate whether or not escape sequences are in use.

Each LEGENDxxx Item can only hold a limited amount of bytes of character data (including the NULL character if any). Consecutive LEGENDxxx Items having the same ItemType can be transmitted to represent longer legend strings.

Within a DF having A661_ASCII_EXTENDED character encoding:

- Each LegendString or ReadoutFormat field must have a NULL terminator

Within a DF having A661_GBK or A661_UTF8 character encoding:

A multi-byte character can span two consecutive LEGENDxxx Items provided they have the same ItemType.

- A NULL terminator is optional for the LegendString or ReadoutFormat field

The LegendString or ReadoutFormat field of a LEGENDxxx Item is followed by zero to three bytes to ensure it terminates on a 32 bit boundary.

A Carriage Return (CR) in a LEGENDxxx is recognized as a normal Carriage Return/Line Feed if the LEGENDxxx follows a LEGEND_ANCHOR. If the

LEGENDxxx follows a SYMBOL, then the meaning of the CR is implementation dependent: it can be interpreted as a normal CR/LF, or it can cause the ensuing text to be rendered at the next desired LEGENDxxx position for that Symbol (i.e., like a "NextField" operation). Interpretation of CR characters in LEGENDS can be controlled by ITEM_STYLE.

This Item has an EndFlag, set to True if it is the last LEGENDxxx Item in a group of consecutive LEGENDxxx Items. When a LEGENDxxx is attached to a Symbol, the LEGENDxxx's EndFlag being set to True also indicates the end of the Symbol Item group.

See Appendix E for an example of attaching a LEGENDxxx to a Symbol.

3.2.10 UA Validation

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

1. CDS will need to know which specific widget instances contain events which will need to be validated by the UA. To support this the UA must set the Enable (A661_ENABLE) parameter to A661_TRUE_WITH_VALIDATION for each applicable widget instance in either the DF and/or during run-time.
2. The UA will need to send a notification to let the CDS know when the UA has completed validating the event. This is accomplished when the UA sends the Run-time Modifiable Parameter A661_ENTRY_VALID.

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

A661_ENTRY_VALID messages should be ignored if the value of the Enable parameter is anything but A661_TRUE_WITH_VALIDATION.

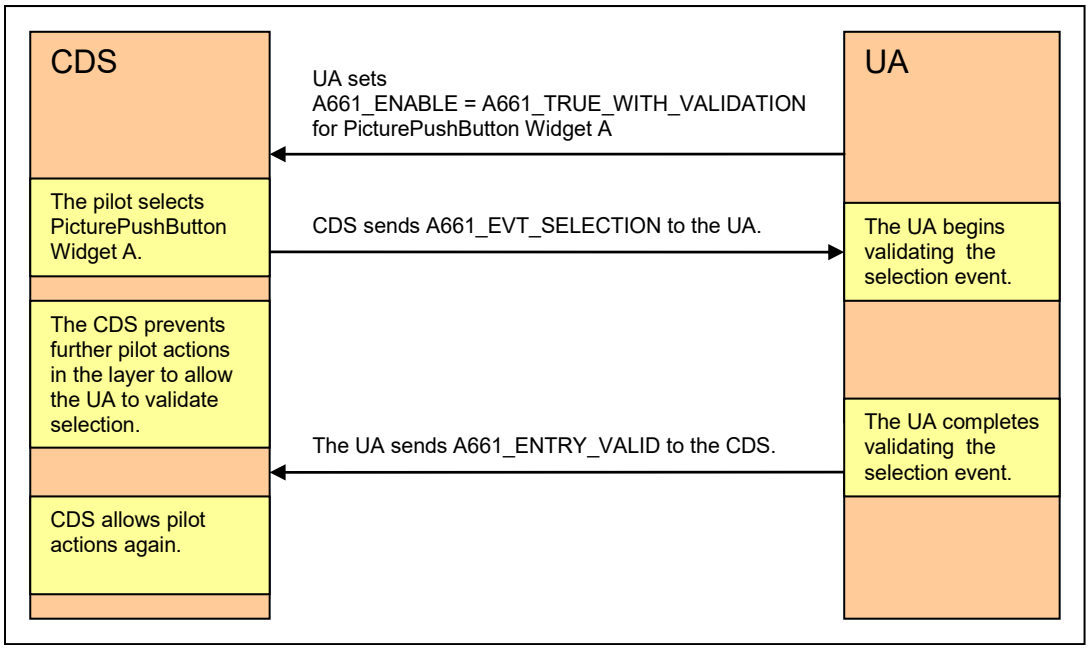


Figure 3.2.10 – Example of UA Validation of a Pilot Selection

The CDS may also implement a timeout period after which it will stop waiting for a UA response. However, there is a risk of losing synchronization with the UA if the CDS receives the entry validation after the timeout expires.

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

- This widget type does not support UA validation. Thus, the setting of A661_ENABLE = A661_TRUE_WITH_VALIDATION will be interpreted by the CDS the same as A661_ENABLE = A661_TRUE. This widget type does not support A661_ENTRY_VALID.
- This widget type supports UA validation and, when the UA has completed validation, the UA will send A661_ENTRY_VALID to the CDS. If during validation the UA determined the pilot action was valid, then A661_ENTRY_VALID will be set to TRUE; if invalid, then set to FALSE.

COMMENTARY

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

One of the motivations behind UA validation is to provide a mechanism to address race conditions (see Section 3.1.2.2).

3.2.11 Widget Extension

Widget Extensions add optional features to existing widgets by adding an extension block to the widget in the definition file. Multiple extension blocks can be applied to a single widget to add multiple optional features to that widget. Runtime modifiable parameters and events may also be associated with an extension. See Section 8 for full details on available extensions.

3.2.12 Animations

Animation widgets provide a means of performing an animation on one or more widget parameters. The widgets on which Animations can be applied must be containers or widgets that have a graphical representation.

The animation state transition algorithm is:

- When the AnimationState parameter is set to A661_PLAY:
  - **The animation effect (the transformation or target parameter, if applicable) is initialized to the "FromValue".**
  - **Then the specified "Delay" is elapsed,**
  - **Then the animation is** progressing to the "ToValue" over time period "Duration".
  - **Note that if the animation is already playing, nothing happens.**

- When the AnimationState parameter is set to A661_ABORT, the animation is aborted immediately **and the animation effect (either transformation or target parameter if applicable) is set to the "FromValue"**. If the animation is not currently playing **this set operation has no effect**.
- **When the CDS causes the animation to abort, it will send an A661_ABORTED event.**
- **When the AnimationState parameter is set to A661_DEACTIVATE (default state at initialization), no animation transformation is applied to the target widget (specific behavior is explained in the specific animations descriptions). This allows for example if you play an animation on a widget, to put back the widget in the graphical state it had before the first time the animation is applied.**
- When the AnimationState parameter is set to A661_END, the animation is **ended** immediately **(even if the AnimationRepetition is set to the Maximum ushort value) and the animation effect (either transformation or target parameter if applicable) is set to the "ToValue".** If the animation is not currently playing this set operation has no effect.
- **When the CDS causes the animation to end, it will send an A661_COMPLETED event.**
- **When the AnimationState parameter is set to A661_FROM, the animation effect (the transformation or target parameter, if applicable) is set to the "FromValue". If the animation is already playing this set operation has no effect.**
- **When the AnimationState parameter is set to A661_TO, the animation effect (the transformation or target parameter, if applicable) is set to the "ToValue". If the animation is already playing this set operation has no effect.**
- **If the "FromValue" for an animation does not correspond to an identity matrix for transformation animations (i.e., 0, 0 for an AnimationTranslation, 0 for AnimationRotation, or 1, 1 for an AnimationScale):**
  - **The widget may appear to jump at the beginning of the animation.**
  - **The behavior of the animation may be different after the first time if has been started if it has not been deactivated (A661_DEACTIVATE) after it has been started.**
- **If the "FromValue" for an animation does not correspond to the current parameter value for AnimationOnParam, the widget may appear to jump at the beginning of the animation.**
- For animations with the AnimationRepetition parameter set to 0 or 1:
  - When the "ToValue" is reached, the translation remains at this value until the animation is started again. The CDS sends the A661_EVT_ANIMATION_STATUS_CHANGE event with the value A661_COMPLETED.
- For animations with the AnimationRepetition parameter set to the maximum value for ushort:

  - When the “ToValue” is reached the Animation starts again from the “FromValue.” The CDS does not send the event A661_EVT_ANIMATION_STATUS_CHANGE.
- For animations with all other values of the AnimationRepetition parameter:
  - When the “ToValue” is reached the Animation starts again from “FromValue” until all the specified repetitions have been performed. At the end of the last repetition, the translation remains at the “ToValue” until the animation is started again. The CDS sends the A661_EVT_ANIMATION_STATUS_CHANGE event with the value A661_COMPLETED.

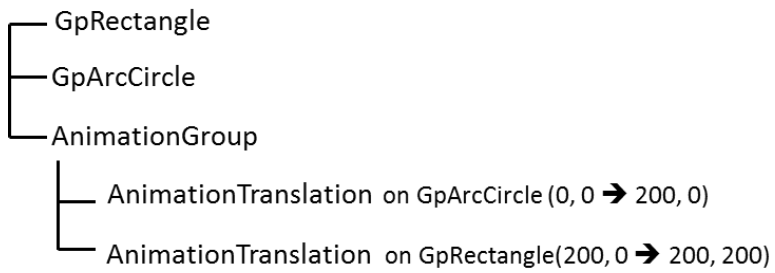
Note that the actual parameter corresponding to the “FromValue” and “ToValue” values depend on the Animation widget. For example, for the AnimationTranslation widget, “FromValue” correspond to the (FromValueX, FromValueY) value of the translation. If an external condition different from the completion of the Animation occurs (for example a cockpit Layout change which results in the disappearance of the Layer which contains the Animation widget), animation widgets which were currently performing an animation may send a A661_EVT_ANIMATION_STATUS_CHANGE with the value A661_ABORTED. The exact conditions of the sending of this event with an A661_ABORTED value are implementation dependent.

Notes:

- **Changing the parameters of an animation during the playing of this animation will have no effect on the current animation**.
- **The Delay for an animation only applies before the start of the animation. If the animation repeats one or more times (AnimationRepetition different from 0 or 1), the Delay will not be applied for each repetition**.
- **When several transformation animations (AnimationScale, AnimationTranslation, AnimationRotation) apply on the same widget, all of the transformation matrices will be combined left to right in the order of the animations in the Definition File. For example. If two AnimationTranslation apply on the same widget, the initialization of the two From values of the animations will be added**.

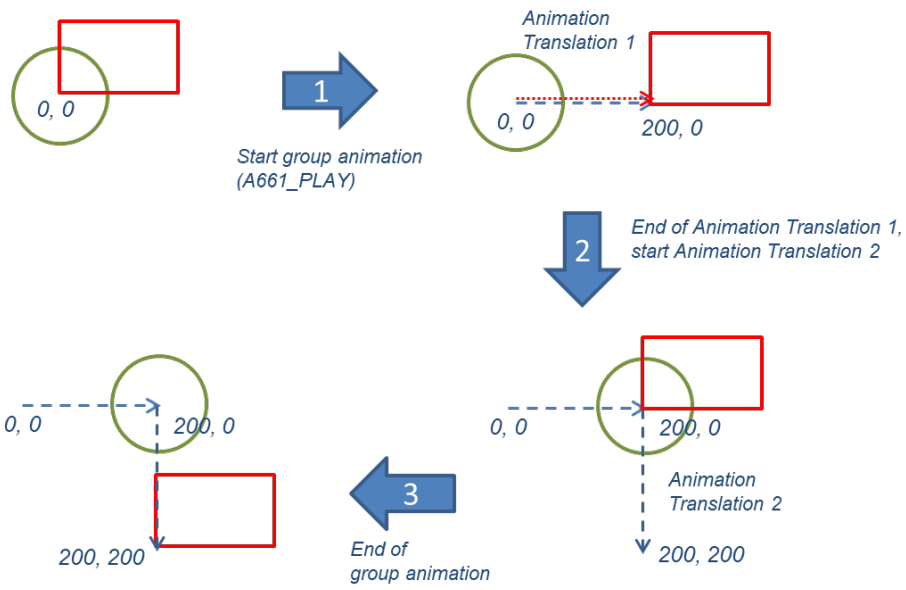
Example with two children animations on two different widgets:

Suppose the following widgets structure:



The first AnimationTranslation applies on the GpArcCircle, the second AnimationTranslation appl on the GpRectangle.

The AnimationGroup Type is set to A661_SEQUENTIAL_FROM, and no repetition nor delay are applied to any animation.



1. **Once state of the AnimationGroup is set to A661_PLAY, the translation of the GpArcCircle is initialized to 0, 0, and the translation of the GpRectangle is initialized to 200, 0. Then the first AnimationTranslation translates the GpArcCircle from 0, 0 to 200, 0,**
2. **When the first AnimationTranslation is ended, the second AnimationTranslation translates the GpRectangle from 200, 0 to 200, 200, while the 200,0 translation on GpArcCircle is maintained.**
3. **When the second AnimationTransation is ended, a 200, 0 translation on the GpArcCircle and a 200,200 translation on the GpRectangle are maintained.**

If the AnimationGroup receives a A661_ABORT before the end of the animation, animation is aborted and the translation applied on GpArcCircle and GpRectangle will correspond to the "FromValue" of applicable AnimationTranslation, so no translation for GpArcCircle and 200,0 translation for GpRectangle (which correspond to the position of the widgets when animation is started).

If the AnimationGroup receives a A661_END before the end of the animation, the animation is ended and the translation applied on GpArcCircle and GpRectangle will correspond to the "ToValue" of applicable AnimationTranslation, so 200,0 translation for GpArcCircle and 200,200 translation for GpRectangle (which correspond to the position of the widgets when animation is ended).

If the AnimationGroup receives a A661_DEACTIVATE whatever the state of the animation, no more translation is applied on GpArcCircle and GpRectangle (which correspond to the position of the widgets before animation was started the first time).

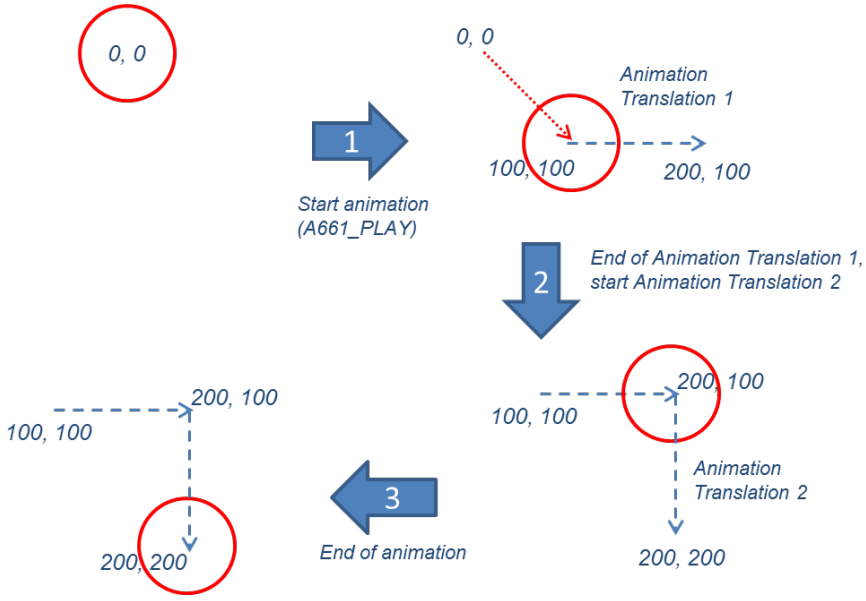
Example with two children animations on the same widget:

Suppose the following widgets structure:

- GpArcCircle
  - AnimationGroup
    - AnimationTranslation on GpArcCircle (100, 0 ➔ 200, 0)
    - AnimationTranslation on GpArcCircle (0, 100 ➔ 0, 200)

The first and second AnimationTranslation both apply on the GpArcCircle.

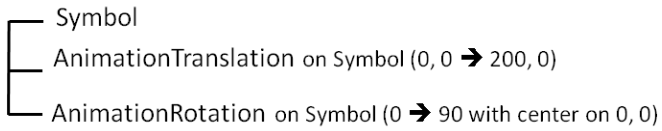
The AnimationGroup Type is set to A661_SEQUENTIAL_FROM, and no repetition nor delay are applied to any animation.



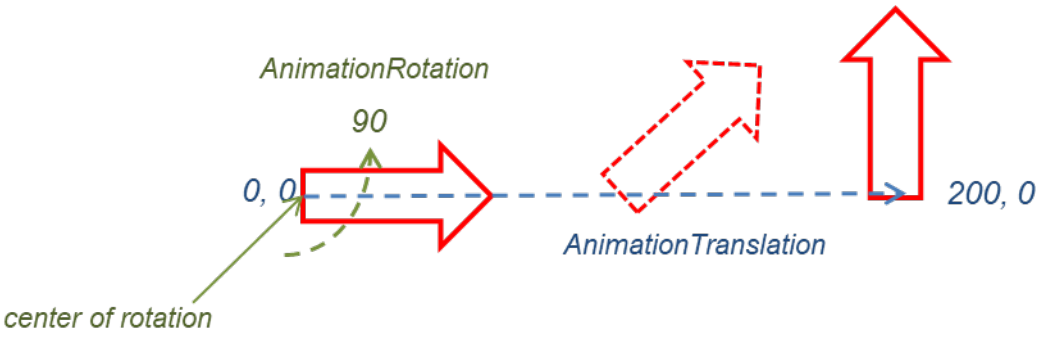
1. **Once state of the AnimationGroup is set to A661_PLAY, the translation of the GpArcCircle is initialized to 100, 100 (which is the combination of the "FromValue" of the 2 AnimationTranslation). Then the first AnimationTranslation translates the GpArcCircle from 100, 100 up to 200, 100 (combination of 0,100 "FromValue" of second AnimationTranslation with current translation value calculated by first AnimationTranslation).**
2. **When the first AnimationTranslation is ended, the second AnimationTranslation translates the GpArcCircle from 200, 100 to 200,200 (combination of 0,100 "ToValue" of first AnimationTranslation with current translation value calculated by second AnimationTranslation).**
3. **When the second AnimationTransation is ended, a 200, 200 translation on the GpArcCircle is maintained (combination of 200, 0 "ToValue" of first AnimationTranslation with 0, 200 "ToValue" of second AnimationTranslation).**

Example of two transformation animations on the same widget:

Suppose the following widgets structure:



The AnimationTranslation and AnimationRotation both apply on the Symbol



If the two animations are started at the same time:

- **The transformation of the Symbol is initialized to a concatenation of a translation of 0, 0 and a rotation of 0, in this order**.
- During the execution of both the animations, the transformation applied to the Symbol is the result of combining the current value of the translation, and the current value of the rotation, in this order.
- **At the end of both the animations, the transformation applied to the Symbol is the result of combining a translation of 200, 0, with a rotation of 90 degrees, in this order**.

3.3 Widget List

This section describes the characteristics and the interface of ARINC 661 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(s)
5. Run-time modifiable parameter table

Each of these parts is described in more detail as follows:

1. The Definition section states the categories of the widget, functional description of the widget and any restrictions to ARINC 661 principles.
2. The Widget Parameters Table 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

Spelling of widget parameter names is consistent across tables. Widget parameter names should not contain spaces. Any spaces appearing in widget parameter names are for document formatting purposes only.

The remaining three parts of each widget description section give the content and format of the exchanges between UA and CDS, the definition-time exchanges as well as the run-time exchanges. These descriptions are completed by information found in Section 4. These descriptions include definitions of data types and encoding. The mapping between Parameter Idents and the type of Set Parameter used is declared in Table 4.6-8, Parameter Types.

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 integer coded on 8 bits
string	Encodes a text string as a sequence of bytes, dependent on the character encoding defined for the Definition File (See Table 4.5.3.2-6 Charset Encoding Block Structure) Ended by NULL character
short	Short integer coded on 16 bits
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 in Two’s Complement Fractional Notation. Signed. Number of significant bits specified in table. Value in parentheses (i.e., ‘x’) represents half the full range. Least Significant Bit (LSB) value is equal to x divided by 2-raised-to-the-number-of-bits-minus-1. To represent angles, x is usually chosen to be 180 degrees. For example, fr(180) in 16 bits is LSB 0.00549316. fr(180) in 32 bits is LSB 8.381903175442e-8. fr(32768) in 32 bits is LSB 0.0000152587890625.
N/A	Non Applicable

All signed numbers are two’s complement form.

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

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

Indices for array structures (StringArray, EntryArray, etc.) should start with the number one (1) unless otherwise indicated.

Unused Pads should be set to zero.

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-bit Word	Name	Type	Size (bits)	Value/Range When Necessary
1	Param1	ushort	16	
	Param2	ushort	16	
2	Param3	ulong	32	
3	Param4	uchar	8	
	Param5	uchar	8	
	Param6	uchar	8	
	Param7	uchar	8	

1. The Creation Structure Table describes the format of the creation structure: i.e., it defines the CreateParameterBuffer structure.
2. 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. The Event Notification Structure Table describes the structure of the events associated with the widget. It describes the events that can be sent to the UA by the CDS. These are usually initiated by a crew member interaction on a widget owned by the UA.
4. The Runtime-Modifiable Parameters Table lists the widget parameters whose values can be changed during run time. This table refers to one or more ParameterStructures, which are usually found in Section 4. This table describes the commands accessible to the UA responsible for managing the widget at runtime.

Some widgets have additional sections to define dedicated data structures.

The following sections define widgets in the Widget Library.

Notation

Notations used in ARINC 661 Command Structures:

Table 3.3-3 – Notations used in ARINC 661 Command Structures

Command	Description
<A>	element of type A
"<A> | <B>"	means <A> or <B>
"{<A>}"	means a set of 0 or more of <A>
"{<A>}+"	means a set of 1 or more of <A>
"(…)"	are used for external references or comments
[<A>]	means zero or one of <A>

3.3.1 ActiveArea

Categories:

- Graphical representation
- Interactive

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

Restriction:
None

ActiveArea Parameters are defined in Table 3.3.1-1.

Table 3.3.1-1 – ActiveArea Parameters

Parameters	Change	Description
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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

ActiveArea Creation Structures are defined in Table 3.3.1-2.

Table 3.3.1-2 – ActiveArea Creation Structure

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

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

Table 3.3.1-3 – ActiveArea Event Structures: A661_EVT_SELECTION

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

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

Table 3.3.1-4 – ActiveArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION

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. The position of the BasicContainer can be changed at run-time.

COMMENTARY

BasicContainer is different from a TranslationContainer because it cannot be the child of a RotationContainer. 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
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's descendants to be interactive
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point

BasicContainer Creation Structure is defined in Table 3.3.2-2.

Table 3.3.2-2 – BasicContainer Creation Structure

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

The BasicContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are **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
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y

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.

BlinkingContainer Creation Structures is defined in Table 3.3.3-2.

Table 3.3.3-2 – BlinkingContainer Creation Structure

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

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
BlinkingType	uchar	8	A661_BLINKING_TYPE

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 parameters modifiable at run-time. The CDS will perform a set on each parameter identified in the structure.

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

Restrictions:

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

A661_ParameterStructure_1Byte
A661_ParameterStructure_2Bytes
A661_ParameterStructure_4Bytes
A661_ParameterStructure_String
A661_ParameterStructure_8Bytes

Variable-size structures cannot be used inside the Buffer parameter of the BufferFormat. The Buffer parameter of the BufferFormat can only be composed of simple parameter values. One exception is the String, which is preceded by two bytes describing the size of the string in bytes (including the NULL character terminating the string).

BufferFormat Parameters are defined in Table 3.3.4-1.

Table 3.3.4-1 – BufferFormat Parameters

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		
NumberOfFields	D	Number of fields in the buffer.
BufferStructure	D	Pairs of WidgetIdent/ParameterIdent for the value to be sent by the UA through the BufferFormat. The number of pairs is defined by the NumberOfFields. The size of this parameter, in bytes, is (WidgetIdent_Size + ParameterIdent_Size)* NumberOfFields
BufferOfParameter	R	Buffer containing the values corresponding to each pair WidgetIdent/ParameterIdent.

BufferFormat Creation Structure is defined in Table 3.3.4-2.

Table 3.3.4-2 – BufferFormat Creation Structure

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

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
BufferOfParameter	N/A	{32}+	A661_BUFFER_OF_PARAM

3.3.4.1 BufferFormat Alignment

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

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

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

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

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

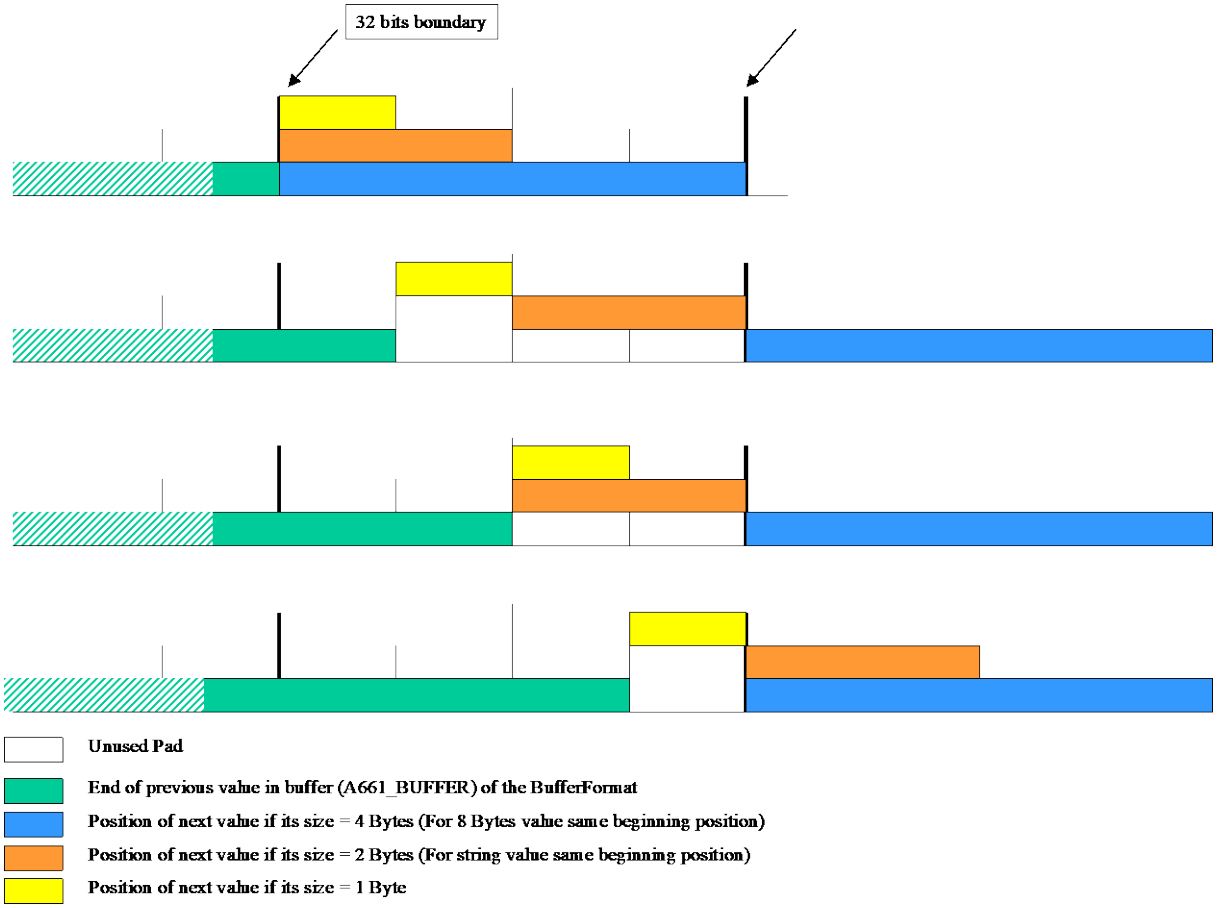


Figure 3.3.4.1-1 – BufferFormat Alignment

3.3.5 CheckButton

Categories:

- Graphical representation
- Interactive
- Text string

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

Restriction:
N/A

CheckButton Parameters are defined in Table 3.3.5-1.

Table 3.3.5-1 – CheckButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CHECK_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
CheckButtonState	DR	Inner state of the CheckButton: SELECTED UNSELECTED
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
LabelString	DR	Label of the CheckButton
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	Alignment of the text within the label area of the widget Left Right Center
PicturePosition	DR	Position of the check box (picture) with respect to the label within the CheckButton Left Right
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

CheckButton Creation Structure is defined in Table 3.3.5-2.

Table 3.3.5-2 – CheckButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CHECK_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	

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

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

Table 3.3.5-3 – CheckButton Event Structures: A661_EVT_STATE_CHANGE

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

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

Table 3.3.5-4 – CheckButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
CheckButtonState	uchar	8	A661_INNER_STATE_CHECK
StyleSet	ushort	16	A661_STYLE_SET
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
LabelString	string	{32}+	A661_STRING
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**

3.3.6 ComboBox

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The ComboBox allows a crew member to choose one entry within a list by interacting with that entry. 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 complete list can also be displayed upon a crew member interaction.

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.

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the ComboBox (in the closed mode)
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaWidth	DR	X Size of the area available to display the popup part (the entry list)
SelectingAreaHeight	DR	Y Size of the area available to display the popup part (the entry list)
OpeningMode	DR	Position of the popup part relative to the static part when it is opened: UP CENTERED DOWN
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	Justification of the text within the label area of the widget LEFT RIGHT CENTER

Parameters	Change	Description
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be less than or equal to MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list.
OpeningEntry	DR	Entry number which is ensured to be visible when the ComboBox is opened. Opening entry is in the range [0; NumberOfEntries] OpeningEntry will be set to 0, if not used.
EntryList	DR	String array holding the list of entries.
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

ComboBox Creation Structure is defined in Table 3.3.6-2.

Table 3.3.6-2 – ComboBox Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_COMBO_BOX
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningEntry	ushort	16	
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
EntryList [NumberOfEntries]	{string}+	{32}+	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.

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

Table 3.3.6-3 – ComboBox Event Structures: A661_EVT_SEL_ENTRY_CHANGE

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

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

Table 3.3.6-4 – ComboBox Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
EntryList [up to MaxNumberOfEntries]	{string}+	{32}+	A661_STRING_ARRAY
OpeningEntry	ushort	16	A661_OPENING_ENTRY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
SelectingAreaWidth	**ulong**	**32**	**A661_SELECTING_WIDTH**
SelectingAreaHeight	**ulong**	**32**	**A661_SELECTING_HEIGHT**
OpeningMode	**uchar**	**8**	**A661_OPENING_MODE**
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.3.7 Connector

Categories:

- Utility

Description:
The purpose of the Connector is to make a layer be the child of a container located in another layer. All of the widgets in the referenced layer effectively become descendants of the Connector's parent and as a result are affected in the same way as any other descendant widgets in an ancestor-descendant relationship, with one

important exception: the Connector can affect the Enable and Visible states of the referenced layer.

Examples of the use of the Connector widget include TabbedPanelGroup and MapHorz, both of which may need to combine display information from several UAs into one integrated presentation. There are other situations where this widget may be used to combine information from different layers as necessary to achieve the desired display presentation.

An example of a parent-child relationship that can be created using a Connector is shown in the diagram below. The CDS will render Layer 7 as if it occurred at the point in Layer 1 where the Connector widget is located.

In this example, UA "A" can affect the Visibility and Enable states of Layer 7 through the Visible and Enable parameters in the Connector. The draw origin of Layer 7 is determined by the draw origin of the Connector's parent (in this case the BasicContainer widget). If the Connector widget had any ancestors that applied clipping (Panels, etc.), then this clipping would also be applied to Layer 7. In all other respects the two layers are independent and the widgets within these respective layers can only be "addressed" by the owning UA.

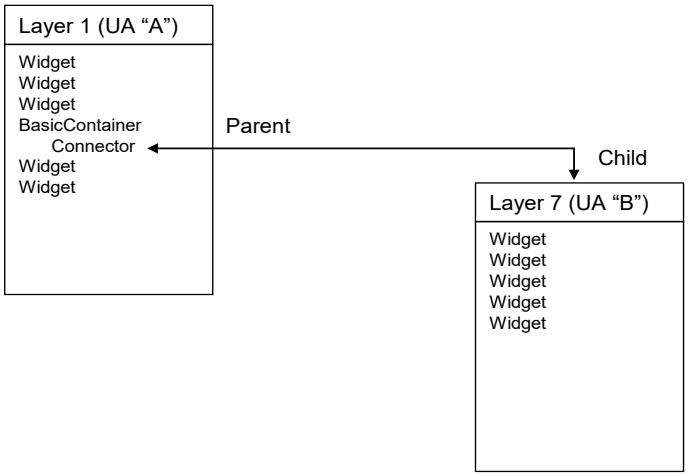


Figure 3.3.7 – Connector Example

Restriction:
The Connector widget does not imply ownership, copying of data, or write access; it only establishes a parent-child relationship. All events associated with the referenced layer are handled by the User Application owning the referenced layer.

The draw priority and clipping of the referenced layer is inherited from its parent Connector.

Except for MapHorz_Source and MapVert_Source, the Connector widget can have any container widget as its parent. It must be ensured that all children in the referenced layer are valid children of the Connector's parent.

COMMENTARY

If two layers, L1 and L2, each own a Connector widget both of which point to the same layer (L3), then L1 and L2 should not be interactive at the same time in a given configuration. If the layer L3 is meant to have a different appearance (e.g., a slightly different layout of the

widgets) in contexts L1 and L2, the layers L1 and L2 should not both be drawn at the same time in a given configuration.

Care should be taken when using the Connector widget, to avoid problems with system certification. This is because the Connector widget provides the means for one UA to affect the graphical properties of widgets owned by another UA through its parent-child relationship. For instance, if UA 1 is developed to DO-178B Level C and connects to UA 3 (which is developed to Level B), then UA 1 has the ability to affect a higher criticality function. This condition would need to be carefully examined to determine its certification acceptability.

Connector Parameters are defined in Table 3.3.7-1.

Table 3.3.7-1 – Connector Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_CONNECTOR
WidgetIdent	D	Unique identifier of the connector
ParentIdent	D	Identifier of the immediate container of the connector
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget's descendants to be interactive
Specific parameters		
Connector Reference	D	Identifier of or Reference to the Connected layer. The resolution of the link between the Connector and the layer is implementation dependent. All events generated by the widgets of the connected (child) layer are still handled by the owning application of that layer.

Connector Creation 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	
ConnectorReference	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
UnusedPad	N/A	16	0

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

The Connector widget does not send any events.

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
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE

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
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	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget

CursorPosOverlay Creation Structure is defined in Table 3.3.8-2.

Table 3.3.8-2 – CursorPosOverlay Creation Structure

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

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

Table 3.3.8-3 – CursorPosOverlay Event Structure: 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
Enable	uchar	8	A661_ENABLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**

3.3.9 EditBoxMasked

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The Masked edit box is similar to the Text edit box except that some characters cannot be modified. The characters that cannot be modified are specified by the UA by setting the “alpha mask” parameter and the “numeric mask” parameters to 0.

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

When the EditBoxMasked is in edit mode, the CDS may report all modifications made to the value of the edited string, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the “ReportAllChanges” parameter.

Restriction:
N/A

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 interactive
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D	The X position of the widget reference point
PosY	D	The Y position of the widget reference point
SizeX	D	The X dimension size (width) of the widget
SizeY	D	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
LabelString	DR	Text of the edit box, this string is limited to 32 characters
StartCursorPos	DR	Start position of cursor in the field when entering edit mode. **See Section 3.1.3.3.**
Alignment	D	Justification of the label text within the edit box area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)

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

EditBoxMasked Creation Structure is defined in Table 3.3.9-2.

Table 3.3.9-2 – EditBoxMasked Creation Structure

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

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

Table 3.3.9-3 – EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE_ABORTED

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

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

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

Table 3.3.9-4 – EditBoxMasked Event Structures: A661_EVT_STRING_CHANGE

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

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

Table 3.3.9-5 – EditBoxMasked Event Structures: A661_EVT_STRING_CONFIRMED

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

Table 3.3.9-6 – EditBoxMasked Event Structures: A661_EVT_EDITBOX_OPENED

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

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

Table 3.3.9-7 – EditBoxMasked Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
LabelString	string	{32}+	A661_STRING
StartCursorPos	ushort	16	A661_CURSOR_POS
StyleSet	ushort	16	A661_STYLE_SET
AlphaMask	ulong	32	A661_ALPHA_MASK
NumericMask	ulong	32	A661_NUMERIC_MASK
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION

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.

When the EditBoxNumeric is in edit mode, the CDS may report all modifications made to the value of the edited string, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the “ReportAllChanges” parameter.

Restriction:
N/A

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
Value	DR	Value displayed by the edit box in normal mode
FormatString	DR	String describing the format of the numeric field. The string is composed of any authorized characters. Some of them will be interpreted to format the value. **See Section 3.2.6 “Formatted Numeric Values.”**
MaxFormatStringLength	D	**Maximum size in bytes (including the NULL terminator) of FormatString.**
TicsCoarse	DR	Coarse increment step for modification of the value with main wheel.
TicsFine	DR	Fine increment step for modification of the value with secondary wheel.
StartCursorPosByte	DR	Start position of cursor in field when entering edit mode. **See Section 3.1.3.3.**
Alignment	DR	Justification of the label text within the edit box area: **A661_**CENTER **A661_**LEFT

Parameters	Change	Description
		A661_RIGHT
ReportAllChanges	D	A661_EDB_CHANGE_CONFIRMED CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) A661_EDB_ALL_CHANGE CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) A661_EDB_OPEN_CLOSE CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
NumericKeyFlag	D	Ability to change the value with the numerical key A661_TRUE A661_FALSE
MaxLegendStringLength	D	Max string size for the legend (MaxLegendStringLength > 0)
LegendAreaSizeX	DR	The X dimension (size) of the legend.
LegendString	DR	Legend associated to the numeric value
LegendPosition	DR	Position of the legend in comparison with the numeric value: A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
LegendRemoved	DR	The flag defining if the legend is to be removed on entry in the editing mode.
MinValue	DR	**Minimum value of the entry (inclusive), applies for edits associated with TicsCoarse and TicsFine.**
MaxValue	DR	**Maximum value of the entry (inclusive), applies for edits associated with TicsCoarse and TicsFine.**
CyclicFlag	D	**Possibility for cyclic modification (or wraparound) of the value on reaching MinValue/MaxValue** **A661_TRUE** **A661_FALSE**

EditBoxNumeric Creation Structure is defined in Table 3.3.10-2.

Table 3.3.10-2 – EditBoxNumeric Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_NUMERIC
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
Value	float	32	
TicsCoarse	float	32	
TicsFine	float	32	
MinValue	float	32	
MaxValue	float	32	
LegendAreaSizeX	ulong	32	
StartCursorPosByte	uchar	8	
MaxFormatStringLength	uchar	8	
MaxLegendStringLength	uchar	8	
LegendPosition	uchar	8	A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_OPEN_CLOSE A661_EDB_ALL_CHANGE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
LegendRemoved	uchar	8	A661_FALSE A661_TRUE
NumericKeyFlag	uchar	8	0
CyclicFlag	uchar	8	
UnusedPad	N/A	16	
FormatString	string	8 * string length	**String terminator NULL is used as string separator before LegendString.**
LegendString	string	8 * string length + PAD	Followed by zero, one, two, or three extra NULL for alignment on 32 bits.

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

Table 3.3.10-3 – EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE_ABORTED

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

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

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

Table 3.3.10-4 – EditBoxNumeric Event Structures: A661_EVT_STRING_CHANGE

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

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

Table 3.3.10-5 – EditBoxNumeric Event Structures: A661_EVT_STRING_CONFIRMED

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

Table 3.3.10-6 – EditBoxNumeric Event Structures: A661_EVT_EDITBOX_OPENED

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

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

Table 3.3.10-7 – EditBoxNumeric Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
EntryValidation	uchar	8	A661_ENTRY_VALID
Value	float	32	A661_VALUE
MinValue MaxValue	float x 2	32 x 2	A661_MINMAX_VALUES
TicsCoarse	float	32	A661_TICS_COARSE
TicsFine	float	32	A661_TICS_FINE
StartCursorPosByte	uchar	8	A661_CURSOR_POS_BYTE
StyleSet	ushort	16	A661_STYLE_SET
LegendString	string	{32}+	A661_STRING
FormatString	string	{32}+	A661_FORMAT_STRING
LegendPosition	uchar	8	A661_LEGEND_POSITION
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
LegendAreaSizeX	**ulong**	**32**	**A661_LEGEND_AREA_SIZE_X**
LegendRemoved	**uchar**	**8**	**A661_LEGEND_REMOVED**

3.3.11 EditBoxText

Categories:

- Graphical representation
- Interactive
- Text string

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

When the EditBoxText is in edit mode, the CDS may report all modifications made to the value of the edited string, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the “ReportAllChanges” parameter.

Restriction:
N/A

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
LabelString	DR	Text of the edit box
StartCursorPos	DR	Start position of cursor in field when entering edit mode. **See Section 3.1.3.3.**
Alignment	DR	Justification of the label text within the edit box area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)

Parameters	Change	Description
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

EditBoxText Creation Structure is defined in Table 3.3.11-2.

Table 3.3.11-2 – EditBoxText Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EDIT_BOX_TEXT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
UnusedPad	N/A	8	0
LabelString	string	8 * string length + 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

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

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

EditBoxText Event Structures: A661_EVT_STRING_CHANGE is 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 is defined in Table 3.3.11-5.

Table 3.3.11-5 – EditBoxText Event Structures: A661_EVT_STRING_CONFIRMED

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

Table 3.3.11-6 – EditBoxText Event Structures: A661_EVT_EDITBOX_OPENED

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

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

Table 3.3.11-7 – EditBoxText Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
EntryValidation	uchar	8	A661_ENTRY_VALID
LabelString	string	{32}+	A661_STRING
StartCursorPos	ushort	16	A661_CURSOR_POS
StyleSet	ushort	16	A661_STYLE_SET
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.3.12 GpArcEllipse

Categories:

- Graphical Representation

Description:

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

Restriction:
None

Figure 3.3.12 – GpArcEllipse

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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of Arc will be filled.
FillIndex	DR	Fill Pattern index. See Section 3.1.3.3.1.
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 A661_SAME_LEVEL
UnusedPad	N/A	16	0

The GpArcEllipse widget does not send any event.

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

Table 3.3.12-3 – GpArcEllipse Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
StartAngle	fr(180)	32	A661_START_ANGLE
EndAngle	fr(180)	32	A661_END_ANGLE
Halo	**uchar**	**8**	**A661_HALO**
Filled	**uchar**	**8**	**A661_FILLED**

3.3.13 GpArcCircle

Categories:

- Graphical Representation

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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of Arc will be filled.
FillIndex	DR	Fill Pattern index. See Section 3.1.3.3.1.
Radius	DR	The radius of the circle
StartAngle	DR	The angle (referenced from the center of the circle) that defines the start position of the arc.
EndAngle	DR	The angle (referenced from the center of the circle) that defines the end position of the arc.

GpArcCircle Creation Structure is defined in Table 3.3.13-2.

Table 3.3.13-2 – GpArcCircle Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_ARC_CIRCLE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
StartAngle	fr(180)	32	
EndAngle	fr(180)	32	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
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 A661_SAME_LEVEL
UnusedPad	N/A	16	0

The GpArcCircle widget does not send any event.

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

Table 3.3.13-3 – GpArcCircle Runtime Modifiable Parameters

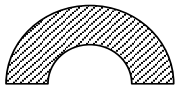
Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Radius	ulong	32	A661_RADIUS
StartAngle	fr(180)	32	A661_START_ANGLE
EndAngle	fr(180)	32	A661_END_ANGLE
Halo	**uchar**	**8**	**A661_HALO**
Filled	**uchar**	**8**	**A661_FILLED**

3.3.14 GpCrown

Categories:

- Graphical Representation

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



CROWN

Restriction:
None

Figure 3.3.14 – GpCrown

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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of Crown will be filled.
FillIndex	DR	Fill Pattern index. See Section 3.1.3.3.1.
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 A661_SAME_LEVEL
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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
InnerRadius	ulong	32	A661_INNER_RADIUS
OuterRadius	ulong	32	A661_OUTER_RADIUS
StartAngle	fr(180)	32	A661_START_ANGLE
EndAngle	fr(180)	32	A661_END_ANGLE
Halo	**uchar**	**8**	**A661_HALO**
Filled	**uchar**	**8**	**A661_FILLED**

3.3.15 GpLine

Categories:

- Graphical Representation

Description:
The graphical primitive GpLine widget enables the definition of a line. The line is defined in rectangular coordinates by two pairs of X and 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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
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 A661_SAME_LEVEL

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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosXStart, PosYStart	long x 2	32 x 2	A661_POS_XY
PosXStart	long	32	A661_POS_X
PosYStart	long	32	A661_POS_Y
PosXEnd, PosYEnd	long x 2	32 x 2	A661_POS_XY2
PosXEnd	long	32	A661_POS_X2
PosYEnd	long	32	A661_POS_Y2
ColorIndex	uchar	8	A661_COLOR_INDEX
Halo	uchar	8	A661_HALO

3.3.16 GpLinePolar

Categories:

- Graphical Representation

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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
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 1/100 millimeters.

GpLinePolar Creation Structure is defined in Table 3.3.16-2.

Table 3.3.16-2 – GpLinePolar Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_LINE_POLAR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosXStart	long	32	
PosYStart	long	32	
RotationAngle	fr(180)	32	
LineLength	ulong	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Halo	uchar	8	A661_FALSE A661_TRUE A661_SAME_LEVEL

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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosXStart, PosYStart	long x 2	32 x 2	A661_POS_XY
PosXStart	long	32	A661_POS_X
PosYStart	long	32	A661_POS_Y
RotationAngle	fr(180)	32	A661_ROTATION_ANGLE
LineLength	ulong	32	A661_LINE_LENGTH
ColorIndex	uchar	8	A661_COLOR_INDEX
Halo	**uchar**	**8**	**A661_HALO**

3.3.17 GpRectangle

Categories:

- Graphical Representation

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. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of Rectangle will be filled.
FillIndex	DR	Fill pattern index. See Section 3.1.3.3.1.

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 A661_SAME_LEVEL
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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Halo	**uchar**	**8**	**A661_HALO**
Filled	**uchar**	**8**	**A661_FILLED**

3.3.18 GpTriangle

Categories:

- Graphical Representation

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**. See Section 3.1.3.3.1**.
Halo	D**R**	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	D**R**	If set to True, interior of Triangle will be filled.
FillIndex	DR	Fill Pattern index**. See Section 3.1.3.3.1.**
PosX2	DR	The X position of the second point of the triangle.
PosY2	DR	The Y position of the second point of the triangle.
PosX3	DR	The X position of the third point of the triangle.
PosY3	DR	The Y position of the third point of the triangle.

GpTriangle Creation Structure is defined in Table 3.3.18-2.

Table 3.3.18-2 – GpTriangle Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_TRIANGLE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
PosX2	long	32	
PosY2	long	32	
PosX3	long	32	
PosY3	long	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid fill index)
Halo	uchar	8	A661_FALSE A661_TRUE A661_SAME_LEVEL
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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX2, PosY2	long x 2	32 x 2	A661_POS_XY2
PosX2	long	32	A661_POS_X2
PosY2	long	32	A661_POS_Y2
PosX3, PosY3	long x 2	32 x 2	A661_POS_XY3
PosX3	long	32	A661_POS_X3
PosY3	long	32	A661_POS_Y3
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Halo	**uchar**	**8**	**A661_HALO**
Filled	**uchar**	**8**	**A661_FILLED**

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.

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	DR	The X position of the widget reference point.
PosY	DR	The Y position of the widget reference point.
SizeX	DR	The X dimension size (width) of the widget.
SizeY	DR	The Y dimension size (height) of the widget.
Specific parameters		
PictureReference	DR	Reference of a picture stored in the CDS.

Picture Creation Structure is defined in Table 3.3.19-2.

Table 3.3.19-2 – Picture Creation Structure

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

The Picture widget does not send any event.

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

Table 3.3.19-3 – Picture Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PictureReference	ushort	16	A661_PICTURE_REFERENCE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**

COMMENT

Image resolution compared to the picture widget size is implementation dependent through the use of the StyleSet parameter.

3.3.20 Label

Categories:

- Graphical representation
- Text string

Description:
A Label widget consists of a text field at a defined display location. If the label is anonymous, it cannot be modified at runtime by the UA. If it is not anonymous, it can be modified by the UA.

Restriction:
None

Label Parameters are defined in Table 3.3.20-1.

Table 3.3.20-1 – Label Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_LABEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Anonymous	D	Ability to be modified at run-time by the UA
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget

Parameters	Change	Description
Specific parameters		
LabelString	DR	Text of the label
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
MotionAllowed	D	Capability to change PosX, PosY, RotationAngle at runtime
RotationAngle	DR	Angle at which LabelString is displayed relative to its origin Refer to Angles defined in Section 2.3.4.2)
Font	DR	Font of the displayed string. **See Section 3.1.3.3.1.**
ColorIndex	DR	Applicable color index for the displayed string. **See Section 3.1.3.3.1.**
Alignment	DR	Justification of the label text within the label area BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight

Label Creation Structure is defined in Table 3.3.20-2.

Table 3.3.20-2 – Label Creation Structure

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

The Label widget does not send any event.

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

Table 3.3.20-3 – Label Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
LabelString	string	{32}+	A661_STRING
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
StyleSet	ushort	16	A661_STYLE_SET
RotationAngle	fr(180)	32	A661_ORIENTATION
ColorIndex	uchar	8	A661_COLOR_INDEX
Font	uchar	8	A661_FONT
Alignment	uchar	8	A661_ALIGNMENT

3.3.21 LabelComplex

Categories:

- Graphical representation
- Text string

Description:
A LabelComplex widget consists of a text field at a defined display location. If the LabelComplex is anonymous, it cannot be modified at runtime by the UA. If it is not anonymous, it can be modified by the UA.

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

Restriction:
N/A

LabelComplex Parameters are defined in Table 3.3.21-1.

Table 3.3.21-1 – LabelComplex Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_LABEL_COMPLEX
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Anonymous	D	Ability to be modified at run-time by the UA
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget

Parameters	Change	Description
Specific parameters		
DefaultStyleText	D	NULL character: Escape sequence not used, default value from the CDS used. “TOutLine⊗TBackColor⊗TForeColor⊗TFont”
LabelString	DR	Text of the label
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	Justification of the label text within the label area BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight

LabelComplex Creation Structure is defined in Table 3.3.21-2.

Table 3.3.21-2 – LabelComplex Creation Structure

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

No event is associated with the LabelComplex widget.

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

Table 3.3.21-3 – LabelComplex Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
LabelString	string	{32}+	A661_STRING
StyleSet	ushort	16	A661_STYLE_SET
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.3.22 MapHorz_ItemList

Categories:

- Map management
- Graphical Representation
- Interactive
- Text string

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

A MapHorz_ItemList must be in a MapHorz_Source container.

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

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

One or several items can be modified through a SetParameter command with BufferOfMapItems or BlockedBufferOfMapItems as Parameter_Ident. An item should be modified in its entirety, for instance the latitude of a symbol item cannot be changed by itself.

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

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

Section 3.3.22.1 describes the standardized items and their functionality.

Section 3.3.22.2 describes the A661_ParameterStructure to address the Items.

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

Table 3.3.22-1 – MapHorz_ItemList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_ITEMLIST
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget.
Enable	DR	Ability of the widget to be interactive.
Specific parameters		
MaxNumberOfItem	D	Maximum number of items that the UA can address under the MapHorz_ItemList.
MaxItemSize	**D**	**Hint provided by the UA that indicates to the CDS what is the expected maximum size in 32-bit words of an item this MapHorz_ItemList may contain.** **If set to 0, no hint is provided.** **Refer to Section 3.3.22.2.1.**
BufferOfMapItems	R	Buffer of the Map Items.
MapSynchronizationNumber	R	See Section 3.2.8.4
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
BlockedBufferOfMapItems	R	Buffer of map items that can span multiple messages in order to make an atomic update to a map item list that cannot fit in a single message.

MapHorz_ItemList Creation Structure is defined in Table 3.3.22-2.

Table 3.3.22-2 – MapHorz_ItemList Creation Structure

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

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

Table 3.3.22-3 – MapHorz_ItemList Event Structures: A661_EVT_SELECTION

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

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

Table 3.3.22-4 – MapHorz_ItemList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
BufferOfMapItems	N/A	{32}	A661_BUFFER_OF_MAPITEM Refer to “MapHorz_ItemList A661_ParameterStructure Specifics”, Section 3.3.22.2 and especially Section 3.3.22.2.2 below.
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4
EntryValidation	uchar	8	A661_ENTRY_VALID
BlockedBufferOfMapItems	N/A	{32}	A661_BLOCKED_BUFFER_OF_MAPITEM Refer to “MapHorz_ItemList A661_ParameterStructure Specifics”, Section 3.3.22.2 and especially Section 3.3.22.2.3 below.

3.3.22.1 MapHorz_ItemList Standard Items Description

This section describes the MapHorz_ItemList items.

Table 3.3.22.1 – MapHorz_ItemList Standard Items Description

Name of Item	Size in Words	Function
BOUNDARY_CHECK	1	This item indicates that boundary checking should be performed on subsequent map items for types to which boundary checking applies (i.e., symbol-based types).
DRAW_LINE_TO_CURSOR	3	This item specifies a point from which a line will be drawn to the current cursor position.
FILLED_POLY_START	3	This Item is used to signify the start of a closed, filled polygon definition. It has X/Y parameters (similar to LINE_START) but also has a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items define the vertices and edges of a polygon that is closed and filled with the indicated fill style. The end of the polygon definition is indicated when the EndFlag in the last LINE_SEGMENT item in the group is set to True.

Name of Item	Size in Words	Function
FILLED_POLY_START_INTERACTIVE	3	This Item is used to signify the start of a closed, filled interactive polygon definition. It has X/Y parameters (similar to LINE_START) but also has a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items define the vertices and edges of a polygon that is closed and filled with the indicated fill style. The end of the polygon definition is indicated when the EndFlag in the last LINE_SEGMENT item in the group is set to True.
ITEM_STYLE	2	For drawing any symbol, line, or text legend the CDS must apply the last defined ITEM_STYLE found in the Map Item List. If no ITEM_STYLE has been provided, the CDS will apply a default ITEM_STYLE. The list of available ITEM_STYLE enumerations and their associated graphical representations are implementation dependent.
LEGEND	5	This Item is used to represent Legend Strings. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_ANCHOR	3	This Item is used to specify the position of a LEGEND not attached to a symbol.
LEGEND_ANCHOR_ROTATED	5	Same as LEGEND_ANCHOR (i.e., Function descriptions for LEGEND_ANCHOR apply to LEGEND_ANCHOR_ROTATED as well) but includes Legend orientation and control parameters.
LEGEND_COMBO	5	This item allows a string to be defined as any combination of LEGEND, LEGEND_POP_UP, and LEGEND_HIGHLIGHT in a single map item. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_HIGHLIGHT	5	This Item is a basic LEGEND, but it will appear only when the associated interactive SYMBOL_x Item is highlighted. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_POP_UP	5	This Item is a basic LEGEND, but it will appear only when the crew member selects the associated SYMBOL_xxx Item. Disappearance of the LEGEND_POP_UP is implementation dependent. Section 3.2.9.5 describes how the LegendString is managed.

Name of Item	Size in Words	Function
LEGEND_READOUT	2	This Item is used to represent a formatted numeric value. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_READOUT_FORMAT_STRING	5	For formatting any LEGEND_READOUT or LEGEND_READOUT_HIGHLIGHT the CDS must apply the last defined LEGEND_READOUT_FORMAT_STRING found in the Map Item List. If no LEGEND_READOUT_FORMAT_STRING has been provided, the CDS will apply a default LEGEND_READOUT_FORMAT_STRING. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_READOUT_COMBO	3	This item allows a string to be defined as any combination of LEGEND_READOUT, popped-up LEGEND_READOUT, and highlighted LEGEND_READOUT in a single map item. Section 3.2.9.5 describes how the LegendString is managed.
LINE_START	3	This Item is used to signify the start of a line. It has only X/Y parameters, interpreted by the CDS depending on the MapHorz_Source DataFormat. The LINE_START, when followed by one or more LINE_SEGMENT and/or LINE_ARC items, defines a line sequence to be rendered. See Appendix E for an example of a Line Item group.
LINE_START_INTERACTIVE	3	This Item is used to signify the start of an interactive line. It has only X/Y parameters, interpreted by the CDS depending on the MapHorz_Source DataFormat. The LINE_START_INTERACTIVE, when followed by one or more LINE_SEGMENT_INTERACTIVE and/or LINE_ARC_INTERACTIVE items, defines a line sequence to be rendered. See Appendix E for an example of a Line Item group.
LINE_SEGMENT	3	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 has an EndFlag, set to True if it is the last item of a Line Item (or polygon) group.

Name of Item	Size in Words	Function
LINE_SEGMENT_INTERACTIVE	**3**	**This Item is used to draw an interactive line, using the last defined style in the list, from the previous LINE_xxx End position, to the specified X/Y coordinates. This Item has an EndFlag, set to True if it is the last item of a Line Item (or polygon) group.**
LINE_ARC	**4**	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 tuple: (InboundCourse, Radius, CourseChange). This Item has an EndFlag, set to True if it is the last item of a Line Item group.
LINE_ARC_INTERACTIVE	**4**	**This Item is used to draw an interactive arc, using the last defined style in the list, from the previous LINE_xxx End position, to the point specified by the tuple: (InboundCourse, Radius, CourseChange). This Item has an EndFlag, set to True if it is the last item of a Line Item group.**
ITEM_SYNCHRONIZATION	**3**	This item has been defined to attach frame data to symbology expressing the context of the computation or the rendering for the symbology frame. This data is sent in ARINC 661 in order to avoid synchronization issues between the symbology frame and the attached information (e.g., mode, range; MRP, etc.). This item can be used to pass synchronization information from the application owning a MapHorz Item List to the application owning the ancestor MapHorz.
NOT_USED	**1**	This Item is used when the Item is to be discarded by the CDS. There is no effect on interpretation of subsequent Items.
PARKING_LINE_START	**5**	Allows the definition of the start of a line which is shown on the map based on a Map Boundary.
PARKING_LINE_END	**4**	Allows the definition of the end of a line which is shown on the map based on a Map Boundary.
SYMBOL_PARKABLE	**5**	Allows defining a symbol that is shown on the map based on a Map Boundary.
SYMBOL_PARKABLE_INTERACTIVE	**5**	Allows defining an interactive symbol that is shown on the map based on a Map Boundary.
SYMBOL_PARKABLE_ROTATED	**6**	**Allows defining a rotated symbol that is shown on the map based on a Map Boundary.**

Name of Item	Size in Words	Function
SYMBOL_PARKABLE_ROTATED_INTERACTIVE	6	Allows defining an interactive rotated symbol that is shown on the map based on a Map Boundary.
SYMBOL_PARKABLE_TARGET	6	Allows defining a target symbol that is shown on the map based on a Map Boundary.
SYMBOL_PARKABLE_TARGET_INTERACTIVE	6	Allows defining an interactive target symbol that is shown on the map based on a Map Boundary.
SYMBOL_GENERIC	4	This Item represents a basic symbol having X/Y position, symbol type and EndFlag parameters. This item has an EndFlag, set to True if no Legend is attached to the Symbol (in effect, this indicates the end of the SYMBOL_xxx Item). If the EndFlag is False, a Legend is to be drawn with the Symbol, and one or more LEGEND items will follow the SYMBOL item. SYMBOL items (and associated LEGENDs, if any) can be embedded in a Line Item group, but not in a Polygon group. See Appendix E for a description of this.
SYMBOL_GENERIC_INTERACTIVE	4	This Item represents a basic interactive symbol having X/Y position, symbol type and EndFlag parameters. This item has an EndFlag, set to True if no Legend is attached to the Symbol (in effect, this indicates the end of the SYMBOL_xxx Item). If the EndFlag is False, a Legend is to be drawn with the Symbol, and one or more LEGEND items will follow the SYMBOL item. SYMBOL items (and associated LEGENDs, if any) can be embedded in a Line Item group, but not in a Polygon group. See Appendix E for a description of this.
SYMBOL_ROTATED	5	Same as SYMBOL_GENERIC (i.e., Function descriptions for SYMBOL_GENERIC apply to SYMBOL_ROTATED as well), except an orientation parameter is added.
SYMBOL_ROTATED_INTERACTIVE	5	Same as SYMBOL_GENERIC_INTERACTIVE (i.e., Function descriptions for SYMBOL_GENERIC_INTERACTIVE apply to SYMBOL_ROTATED_INTERACTIVE as well), except an orientation parameter is added.

Name of Item	Size in Words	Function
SYMBOL_CIRCLE	4	Specific Symbol. It represents a circle of specific radius. Radius is expressed in nm. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.
SYMBOL_CIRCLE_INTERACTIVE	4	**Specific interactive symbol. It represents a circle of specific radius. Radius is expressed in nm. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.**
SYMBOL_OVAL	5	Specific symbol. It represents an oval, filled with the indicated fill style, which may be “no fill.”
SYMBOL_OVAL_INTERACTIVE	5	**Specific interactive symbol. It represents an oval, filled with the indicated fill style, which may be “no fill.”**
SYMBOL_RUNWAY	5	Specific Symbol. It represents an airport runway, with runway orientation and length parameters. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.
SYMBOL_RUNWAY_INTERACTIVE	5	**Specific interactive symbol. It represents an airport runway, with runway orientation and length parameters. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.**
SYMBOL_TARGET	5	Same as SYMBOL_GENERIC (i.e., Function descriptions for SYMBOL_GENERIC apply to SYMBOL_TARGET as well), except part of the symbol can be rotated and part of the symbol has variable length illustrating Velocity / Distance / Altitude.
SYMBOL_TARGET_INTERACTIVE	5	**Same as SYMBOL_GENERIC_INTERACTIVE (i.e., Function descriptions for SYMBOL_GENERIC_INTERACTIVE apply to SYMBOL_TARGET_INTERACTIVE as well), except part of the symbol can be rotated and part of the symbol has variable length illustrating Velocity / Distance/Altitude.**
SYMBOL_RECTANGLE	5	Specific symbol. It represents a rectangle, filled with the indicated fill style, which may be “no fill.”
SYMBOL_RECTANGLE_INTERACTIVE	5	**Specific interactive symbol. It represents a rectangle, filled with the indicated fill style, which may be “no fill.”**

Name of Item	Size in Words	Function
TRIANGLE_STRIP_START	5	This Item is used to signify the start of a closed, filled polygon defined by a series of triangles arranged in a strip.
TRIANGLE_STRIP_START_INTERACTIVE	5	This Item is used to signify the start of a closed, filled interactive polygon defined by a series of triangles arranged in a strip.
TRIANGLE_FAN_START	5	This Item is used to signify the start of a closed, filled polygon defined by a series of triangles arranged in a fan.
TRIANGLE_FAN_START_INTERACTIVE	5	This Item is used to signify the start of a closed, filled interactive polygon defined by a series of triangles arranged in a fan.
TRIANGLE_SEGMENT	3	Defines a single vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_INTERACTIVE	3	Defines a single vertex of an interactive Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_DOUBLE	5	Defines two vertices of a Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE	5	Defines two vertices of an interactive Triangle Strip or Triangle Fan.
TRIANGLE_END	3	Defines the last vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_END_INTERACTIVE	3	Defines the last vertex of an interactive Triangle Strip or Triangle Fan.
TRIANGLE_END_DOUBLE	5	Defines the last two vertices of Triangle Strip or Triangle Fan.
TRIANGLE_END_DOUBLE_INTERACTIVE	5	Defines the last two vertices of an interactive Triangle Strip or Triangle Fan.
VERTEX_RENDER_START	5	Renders geometry from vertices stored in a MapHorz_VertexBuffer.
VERTEX_RENDER_SEGMENT	5	Used following a Vertex_Render_Start to continue referencing indices from the MapHorz_VertexBuffer.
VERTEX_RENDER_END	5	Final element following a Vertex_Render_Start or Vertex_Render_Segment to indicate the end of the rendered geometry.

Error Handling For MIL Item EndFlag

The preceding table describes how the EndFlag should be used by the UA when constructing MapItem Lists. Associated error detection and handling (e.g., a missing EndFlag) is implementation dependent.

Positioning Of Map Symbology

For SYMBOL_xxx map items, the X/Y coordinate parameters provided by the UA constitute a symbol positioning reference point. Generally, this point is expressed in real world coordinates. The OEM and CDS developer must agree on how the CDS should position the corresponding graphics symbols on the map display. For some map items it will be appropriate to display the symbol so it is centered on the real

world position represented by the X/Y reference point. For other items, the CDS will place symbols based on the functional meaning of the map item (e.g., when using SYMBOL_RUNWAY the UA will usually set the X/Y reference point to the airport runway threshold position). The SymbolType parameter and ITEM_STYLE are also used to control positioning of symbols on the map display.

Relative Positioning Of Map Symbology

Placement of map items is generally determined using a real world coordinate system. In order to support decoration of symbols, Relative Positioning is also available for several item types. This allows graphics items to be placed on a map relative to a Symbol or some other fixed location, using screen coordinates instead of real world coordinates. The positioning of these relative elements does not change based on the parameters used to calculate coordinate transformations. Map Item List Items that have the RelativePosition parameter can be positioned via Relative Positioning.

Example of SYMBOL and LEGEND placement using relative screen coordinates:

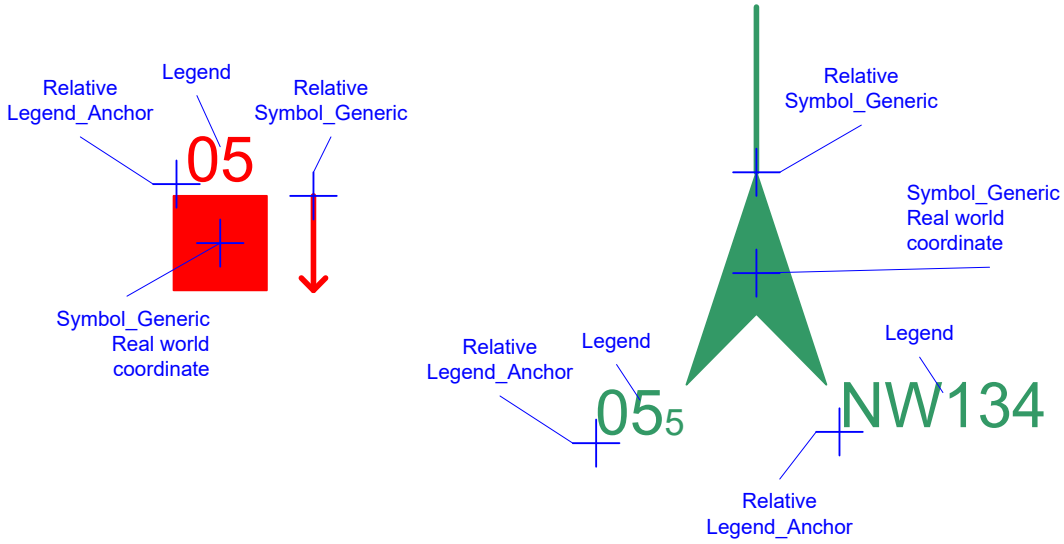


Figure 3.3.22.1 – Examples of Symbol Placement With Relative Screen Coordinates

Overlapping Active Areas

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

3.3.22.2 MapHorz_ItemList A661_ParameterStructure Specifics

This section describes the A661_ParameterStructure_BufferOfItems for MapHorz_ItemList.

3.3.22.2.1 Item Structures

This section describes the MIL 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.

COMMENTARY

CDS implementers will want to bound the amount of memory used by the Map Item List. This could be determined either by observing that there is a maximum size for any MIL item **or by using the MaxItemSize parameter**.

Maximum size for any MIL can be deduced from **Table 3.3.22.1.** The memory required by the CDS to store the MIL could be the value of the MaxNumberOfItem parameter in the MapHorz_ItemList creation structure multiplied by the current maximum MIL item size.

The UA can also propose a MaxItemSize parameter. This parameter can be used to specify the actual maximum item size for a MIL or any optimized size for the map item. (average size, average size with margin, etc.). The memory required by the CDS to store the MIL could be the value of the MaxNumberOfItem parameter in the MapHorz_ItemList creation structure multiplied by the MaxItemSize parameter.
In case a larger item is received, CDS behavior is implementation dependent.
In case the parameter is set to 0, it is up to the CDS to manage memory allocation.

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)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_STYLE
UnusedPad	N/A	8	0
ItemStyleSet	ushort	16	
UnusedPad	N/A	16	0

3.3.22.2.1.2 Legend_Anchor

Legend_Anchor is defined in Table 3.3.22.2.1.2.

Table 3.3.22.2.1.2 – Legend_Anchor

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b

3.3.22.2.1.3 Legend, Legend_Highlight, and Legend_Pop_Up

Legend, Legend_Highlight, and Legend_Pop_Up are 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)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND A661_LEGEND_HIGHLIGHT A661_LEGEND_POP_UP
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendString	{uchar}+ (not 'string' **because it might not be null terminated**)	{32}+	**Maximum of 16 bytes of character data. Section 3.2.9.5 defines the proper string termination and padding.**

3.3.22.2.1.4 Line_Start and Line_Start_Interactive

Line_Start **and Line_Start_Interactive** is defined in Table 3.3.22.2.1.4.

Table 3.3.22.2.1.4 – Line_Start and Line_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_START **A661_LINE_START_INTERACTIVE**
UnusedPad	N/A	8	0
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b

3.3.22.2.1.5 Line_Segment and Line_Segment_Interactive

Line_Segment **and Line_Segment_Interactive** is defined in Table 3.3.22.2.1.5.

Table 3.3.22.2.1.5 – Line_Segment and Line_Segment_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_SEGMENT **A661_LINE_SEGMENT_INTERACTIVE**
EndFlag	uchar	8	A661_TRUE A661_FALSE
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b

3.3.22.2.1.6 Line_Arc and Line_Arc_Interactive

Line_Arc and Line_Arc_Interactive is defined in Table 3.3.22.2.1.6.

Table 3.3.22.2.1.6 – Line_Arc and Line_Arc_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_ARC A661_LINE_ARC_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
InboundCourse	fr(180)	32	A positive InboundCourse indicates a clockwise inbound course with respect to true north (angle∈ [0;180]). A negative InboundCourse indicates a counter-clockwise inbound course with respect to true north (angle= |InboundCourse| ∈ [0;180]). See figure that follows this table for more information.
Radius	fr(32768)	32	Turn radius Positive for clockwise turn (turn right), Negative for counter-clockwise turn (turn left) in nautical miles.
CourseChange	fr(180)	32	A positive CourseChange indicates a turn (Right or left depending of radius sign) of less than 180 degrees: the resulting angle is equal to: CourseChange ∈ [0;180]. A negative CourseChange indicates a turn (Right or left depending of radius sign) of more than 180 degrees: the resulting angle is equal to: (360 + CourseChange) ∈ [0;180].

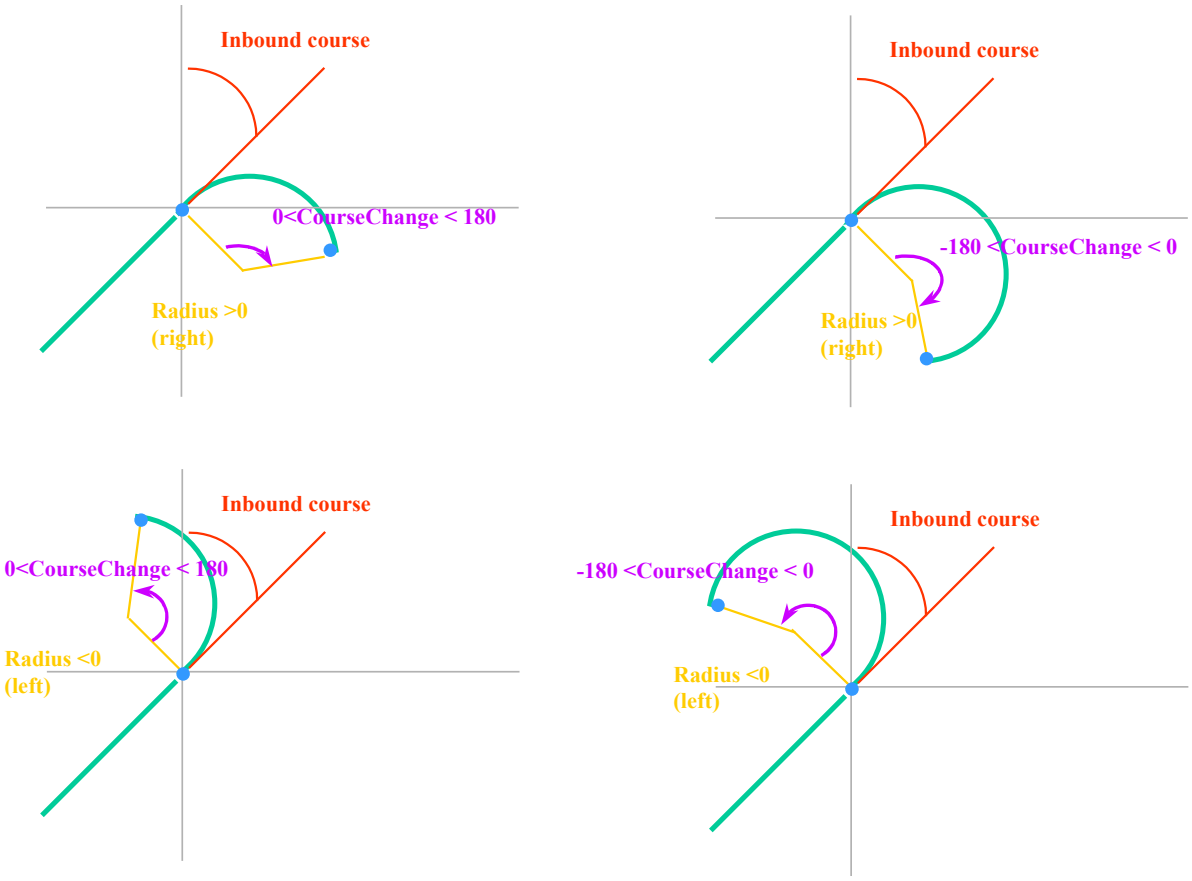


Figure 3.3.22.2.1.6 – Line_Arc Examples

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

3.3.22.2.1.7 Not_Used

Not_Used is defined in Table 3.3.22.2.1.7.

Table 3.3.22.2.1.7 – Not_Used

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

3.3.22.2.1.8 Symbol_Generic and Symbol_Generic_Interactive

Symbol_Generic and Symbol_Generic_Interactive is defined in Table 3.3.22.2.1.8. The symbol representation's bounding circle can be used to define the extent for visibility-testing purposes; see the BOUNDING_CIRCLE command in Section 5.2.4.15.

Table 3.3.22.2.1.8 – Symbol_Generic and Symbol_Generic_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_GENERIC A661_SYMBOL_GENERIC_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b

3.3.22.2.1.9 Symbol_Circle and Symbol_Circle_Interactive

Symbol_Circle and Symbol_Circle_Interactive is defined in Table 3.3.22.2.1.9.

Table 3.3.22.2.1.9 – Symbol_Circle and Symbol_Circle_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_CIRCLE A661_SYMBOL_CIRCLE_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b
Radius	fr(32768)	32	Radius of circle, in nautical miles.

3.3.22.2.1.10 Symbol_Rotated and Symbol_Rotated_Interactive

Symbol_Rotated and Symbol_Rotated_Interactive is defined in Table 3.3.22.2.1.10. The symbol representation's bounding circle can be used to define the extent for visibility-testing purposes; see the BOUNDING_CIRCLE command in Section 5.2.4.15.

Table 3.3.22.2.1.10 – Symbol_Rotated and Symbol_Rotated_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_ROTATED A661_SYMBOL_ROTATED_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_HOLD_LEFT SYMBOL_HOLD_RIGHT SYMBOL_PROCEDURE_TURN_LEFT SYMBOL_PROCEDURE_TURN_RIGHT SYMBOL_LONG_RANGE_AIRPORT_WITH_RUNWAY
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b
Orientation	fr(180)	32	Orientation of Symbol (clockwise is positive rotation) relative to True North

3.3.22.2.1.11 Symbol_Runway and Symbol_Runway_Interactive

Symbol_Runway and Symbol_Runway_Interactive is defined in Table 3.3.22.2.1.11.

Table 3.3.22.2.1.11 – Symbol_Runway and Symbol_Runway_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RUNWAY A661_SYMBOL_RUNWAY_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b
Length	fr(32768)	32	Length of runway (in feet)
Orientation	fr(180)	32	Orientation of Symbol (clockwise is positive rotation) relative to True North

3.3.22.2.1.12 Filled_Poly_Start and Filled_Poly_Start_Interactive

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 notification. The OEM free data field may include, for example, the ItemIndex to identify the error.

Filled_Poly_Start and Filled_Poly_Start_Interactive is defined in Table 3.2.22.2.1.12.

Table 3.2.22.2.1.12 – Filled_Poly_Start and Filled_Poly_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_FILLED_POLY_START A661_FILLED_POLY_START_INTERACTIVE
FillStyleIndex	uchar	8	See Section 3.1.3.3.1
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b

3.3.22.2.1.13 Symbol_Oval and Symbol_Oval_Interactive

Symbol_Oval and Symbol_Oval_Interactive is defined in Table 3.3.22.2.1.13.

Table 3.3.22.2.1.13 – Symbol_Oval and Symbol_Oval_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_OVAL A661_SYMBOL_OVAL_INTERACTIVE
FillStyleIndex	uchar	8	See Section 3.1.3.3.1
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b
Radius	fr(32768)	32	Half the Major Axis in nautical miles
AxisRatio	fr(1)	16	Minor Axis divided by Major Axis
Orientation	fr(180)	16	Orientation of Major Axis (clockwise is positive rotation) relative to True North

3.3.22.2.1.14 Item_Synchronization

Item_Synchronization is defined in Table 3.3.22.2.1.14.

Table 3.3.22.2.1.14 – Item_Synchronization

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

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

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

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

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

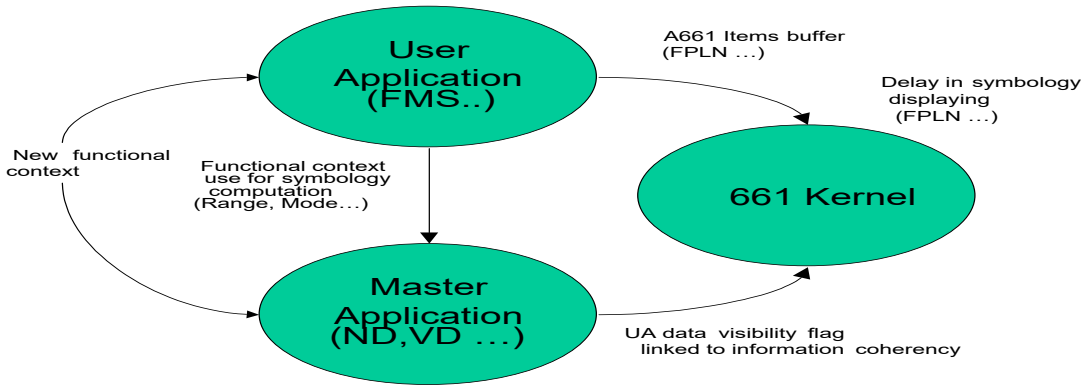


Figure 3.3.22.2.1.14-1 – Map Management Without Item Synchronization

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

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

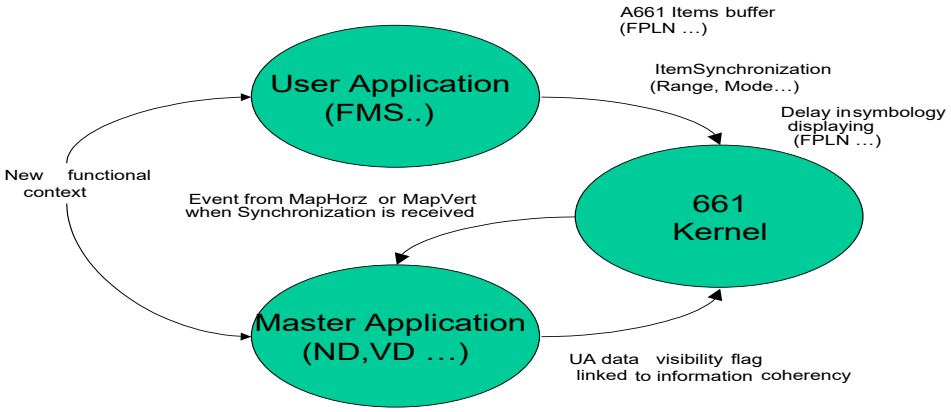


Figure 3.3.22.2.1.14-2 – Map Management With Item Synchronization

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

3.3.22.2.1.15 Symbol_Target and Symbol_Target_Interactive

Symbol_Target and Symbol_Target_Interactive is defined in Table 3.3.22.2.1.15.

Table 3.3.22.2.1.15 – Symbol_Target and Symbol_Target_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_TARGET A661_SYMBOL_TARGET_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: AIR_FRIEND_TRACK AIR_SUSPECT_TRACK
Length	ushort	16	Velocity/Distance/Altitude illustrated by variable length part of the symbol (Knots/Nautical Miles/Feet). The choice of units is an OEM decision. The units for the MIL instance can be determined by the value of the SymbolType parameter, or in some other way.
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b
Orientation	fr(180)	32	Orientation of rotated part (clockwise is positive rotation) relative to True North

An example of how Symbol Target could be used is seen in Figure 3.3.22.2.1.15.

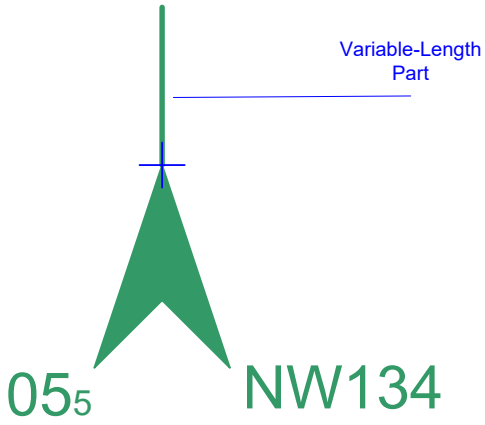


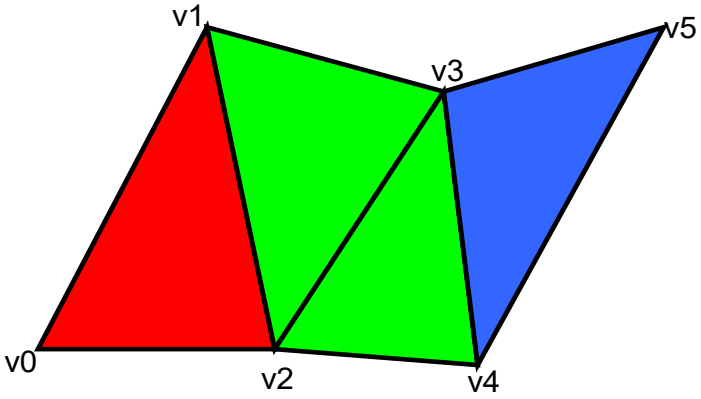
Figure 3.3.22.2.1.15 – Symbol Target Example

3.3.22.2.1.16 Triangle_Strip_Start and Triangle_Strip_Start_Interactive

The Triangle_Strip_Start, Triangle_Segment, Triangle_Segment_Double, Triangle_End, and Triangle_End_Double MapItemList Items are meant to define triangle strips, similar to those defined in SDL Symbols.

Each Triangle Strip is made up of one Triangle_Strip_Start, any number of Triangle_Segment and Triangle_Segment_Double items, and one Triangle_End or Triangle_End_Double. The Triangle_Segment_Double and Triangle_End_Double items exist to minimize MapItemList size and should be used whenever possible. See the illustration below for more details.

NOTE: Lines shown for illustration only



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

Figure 3.3.22.2.1.16 – Triangle Strip

Triangle_Strip_Start **and Triangle_Strip_Start_Interactive** is defined in Table 3.3.22.2.1.16.

Table 3.3.22.2.1.16 – Triangle_Strip_Start Triangle_Strip_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_STRIP_START **A661_TRIANGLE_STRIP_START_INTERACTIVE**
UnusedPad	N/A	8	0
X1	Dependent on MapDataFormat	32	First coordinate of first vertex. See Table 3.3.24-2b
Y1	Dependent on MapDataFormat	32	Second coordinate of first vertex. See Table 3.3.24-2b
X2	Dependent on MapDataFormat	32	First coordinate of second vertex. See Table 3.3.24-2b
Y2	Dependent on MapDataFormat	32	Second coordinate of second vertex. See Table 3.3.24-2b

3.3.22.2.1.17 Triangle_Segment and Triangle_Segment_Interative

The fill of the triangle completed by this item is defined by the FillStyleIndex parameter.

Triangle_Segment **and Triangle_Segment_Interative** is defined in Table 3.3.22.2.1.17.

Table 3.3.22.2.1.17 – Triangle_Segment and Triangle_Segment_Interative

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_SEGMENT **A661_TRIANGLE_SEGMENT_INTERACTIVE**
FillStyleIndex	uchar	8	**Fill for the triangle completed by this point see Section 3.1.3.3.1**
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b

3.3.22.2.1.18 Triangle_Segment_Double and Triangle_Segment_Double_Interactive

The fill of the triangles completed by this item is defined by the FillStyleIndex parameter.

Triangle_Segment_Double and Triangle_Segment_Double_Interactive is defined in Table 3.3.22.2.1.18.

Table 3.3.22.2.1.18 – Triangle_Segment_Double and Triangle_Segment_Double_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_SEGMENT_DOUBLE A661_TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangles completed by these points see Section 3.1.3.3.1.
X1	Dependent on MapDataFormat	32	First coordinate of first vertex. See Table 3.3.24-2b.
Y1	Dependent on MapDataFormat	32	Second coordinate of first vertex. See Table 3.3.24-2b.
X2	Dependent on MapDataFormat	32	First coordinate of second vertex. See Table 3.3.24-2b.
Y2	Dependent on MapDataFormat	32	Second coordinate of second vertex. See Table 3.3.24-2b.

3.3.22.2.1.19 Triangle_End and Triangle_End_Interactive

The fill of the triangle completed by this item is defined by the FillStyleIndex parameter.

Triangle_End and Triangle_End_Interactive is defined in Table 3.3.22.2.1.19.

Table 3.3.22.2.1.19 – Triangle_End and Triangle_End_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_END A661_TRIANGLE_END_INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangle completed by this point see Section 3.1.3.3.1.
X	Dependent on MapDataFormat	32	First coordinate of vertex. See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	Second coordinate of vertex. See Table 3.3.24-2b.

3.3.22.2.1.20 Triangle_End_Double and Triangle_End_Double_Interactive

The fill of the triangles completed by this item is defined by the FillStyleIndex parameter.

Triangle_End_Double and Triangle_End_Double_Interactive is defined in Table 3.3.22.2.1.20.

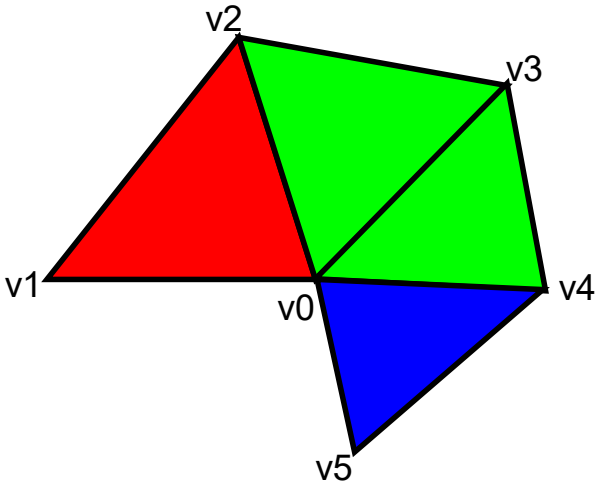
Table 3.3.22.2.1.20 – Triangle_End_Double and Triangle_End_Double_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_END_DOUBLE A661_TRIANGLE_END_DOUBLE_INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangles completed by these points see Section 3.1.3.3.1.
X1	Dependent on MapDataFormat	32	First coordinate of first vertex. See Table 3.3.24-2b.
Y1	Dependent on MapDataFormat	32	Second coordinate of first vertex. See Table 3.3.24-2b.
X2	Dependent on MapDataFormat	32	First coordinate of second vertex. See Table 3.3.24-2b.
Y2	Dependent on MapDataFormat	32	Second coordinate of second vertex. See Table 3.3.24-2b.

3.3.22.2.1.21 Triangle_Fan_Start and Triangle_Fan_Start_Interactive

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

NOTE: Black lines shown for illustration only



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

Figure 3.3.22.2.1.21 – Triangle Fan

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

Triangle_Fan_Start **and Triangle_Fan_Start_Interactive** is defined in Table 3.3.22.2.1.21.

Table 3.3.22.2.1.21 – Triangle_Fan_Start and Triangle_Fan_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_FAN_START **A661_TRIANGLE_FAN_START_INTERFACE**
UnusedPad	N/A	8	0
X1	Dependent on MapDataFormat	32	First coordinate of first vertex. See Table 3.3.24-2b.
Y1	Dependent on MapDataFormat	32	Second coordinate of first vertex. See Table 3.3.24-2b.
X2	Dependent on MapDataFormat	32	First coordinate of second vertex. See Table 3.3.24-2b.
Y2	Dependent on MapDataFormat	32	Second coordinate of second vertex. See Table 3.3.24-2b.

Triangle_Segment, Triangle_Segment_Double, Triangle_End, and Triangle_End_Double are already defined (this was done in conjunction with Triangle_Strip_Start).

The CDS will process subsequent Triangle_Segment, Triangle_Segment_Double, Triangle_End, and Triangle_End_Double items appropriately based on the type of triangle start item that preceded them. That is, if a Triangle_Fan_Start item begins a sequence of triangle items, the first and third vertex of the previous triangle is used as the base for the triangle that follows. If a Triangle_Strip_Start item begins a sequence of triangle items, the second and third vertex of the previous triangle is used as the base for the triangle that follows.

3.3.22.2.1.22 Legend_Anchor_Rotated

Legend_Anchor_Rotated allows the UA to draw a legend on a map at a fixed orientation with respect to True North.

Note: The alignment of the text (its position relative to the Legend_Anchor_Rotated X/Y point) can be defined by the UA through the ITEM_STYLE Item.

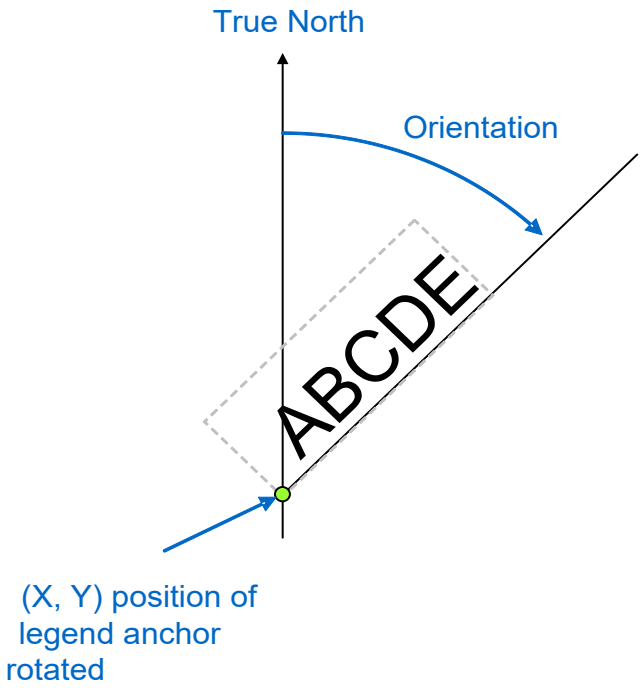


Figure 3.3.22.2.1.22 – Legend_Anchor_Rotated

Legend_Anchor_Rotated is defined in Table 3.3.22.2.1.22.

Table 3.3.22.2.1.22 – Legend_Anchor_Rotated

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR_ROTATED
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b.
Orientation	fr(180)	32	Orientation of Legend (clockwise is positive rotation) relative to True North.

Name	Type	Size (bits)	Description and Value/Range
LegendFlip	uchar	8	A661_FALSE A661_TRUE Defines whether the legend is flipped over when the map rotation makes the text inverted.
UnusedPad	N/A	24	0

3.3.22.2.1.23 Draw_Line_To_Cursor

Draw_Line_To_Cursor allows the UA to specify a point from which a line will be drawn to the current location of the cursor. The appearance of the line can be defined by the UA through the ITEM_STYLE Item.

In the following example, the UA has specified a point coincident with the LISBO waypoint (e.g., the X/Y parameters of the Draw_Line_To_Cursor map item match the X/Y parameters of the map item representing the LISBO waypoint), and the CDS has drawn a line from that point to the cursor.

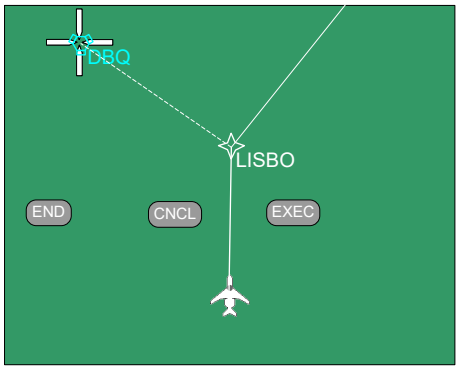


Figure 3.3.22.2.1.23 – Draw_Line_To_Cursor Example

The CDS behavior when there are multiple cursors inside the associated MapHorz display area is implementation dependent.

Draw_Line_To_Cursor is defined in Table 3.3.22.2.1.23.

Table 3.3.22.2.1.23 – Draw_Line_To_Cursor

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_DRAW_LINE_TO_CURSOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b.

3.3.22.2.1.24 Symbol_Rectangle and Symbol_Rectangle_Interactive

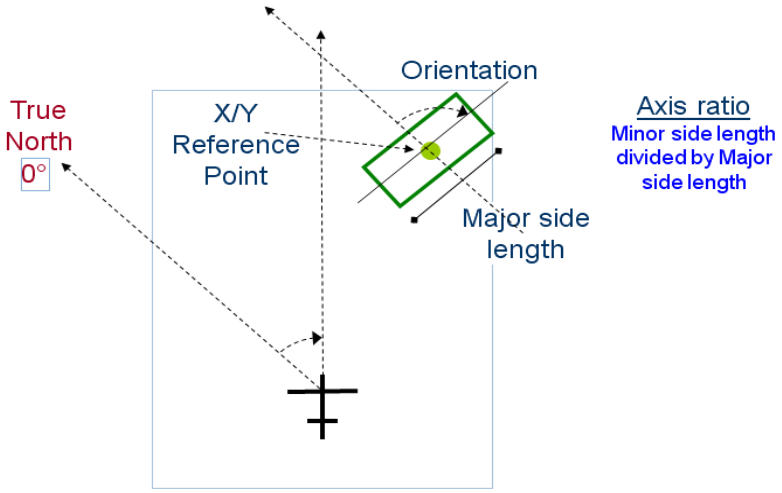


Figure 3.3.22.2.1.24 – SYMBOL_RECTANGLE Example

Symbol_Rectangle and Symbol_Rectangle_Interactive is defined in Table 3.3.22.2.1.24.

Table 3.3.22.2.1.24 – Symbol_Rectangle and Symbol_Rectangle_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RECTANGLE A661_SYMBOL_RECTANGLE_INTERACTIVE
FillStyleIndex	uchar	8	See Section 3.1.3.3.1.
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b.
MajorSideLength	fr(32768)	32	Length of the Major Side in nautical miles.
AxisRatio	fr(1)	16	Minor Side length divided by Major Side Length. Must be greater than zero.
Orientation	fr(180)	16	Orientation of Major Side (clockwise is positive rotation) relative to True North.

3.3.22.2.1.25 Legend_Combo

Legend_Combo allows a string to be defined as any combination of LEGEND, LEGEND_POP_UP, and LEGEND_HIGHLIGHT in a single map item. This allows fewer maps items to be used to define the same behavior.

LegendType	Equivalent to
A661_LEGEND_ONLY	One A661_LEGEND item
A661_POPUP_ONLY	One A661_LEGEND_POP_UP item
A661_HIGHLIGHT_ONLY	One A661_LEGEND_HIGHLIGHT item
A661_LEGEND_AND_POPUP	One A661_LEGEND item One A661_LEGEND_POP_UP item With the same String
A661_LEGEND_AND_HIGHLIGHT	One A661_LEGEND item One A661_LEGEND_HIGHLIGHT item With the same String
A661_POPUP_AND_HIGHLIGHT	One A661_LEGEND_POP_UP item One A661_LEGEND_HIGHLIGHT item With the same String
A661_LEGEND_AND_POPUP_AND_HIGHLIGHT	One A661_LEGEND item One A661_LEGEND_POP_UP item One A661_LEGEND_HIGHLIGHT item With the same String

Figure 3.3.22.2.1.25 – Legend_Combo

Legend_Combo is defined in Table 3.3.22.2.1.25.

Table 3.3.22.2.1.25-2 – Legend_Combo

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_COMBO
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendTypes	uchar	8	A661_LEGEND_ONLY A661_POPUP_ONLY A661_HIGHLIGHT_ONLY A661_LEGEND_AND_POPUP A661_LEGEND_AND_HIGHLIGHT A661_POPUP_AND_HIGHLIGHT A661_LEGEND_AND_POPUP_AND_HIGHLIGHT Value of 0 is reserved
LegendString	{uchar}+ (not 'string' because it might not be null terminated)	{8}+	Maximum of 15 characters. Section 3.2.9.5 defines the proper string termination and padding.

3.3.22.2.1.26 Vertex_Render_Start

The Vertex_Render_Start MapItem enables rendering of the vertices contained in a MapHorz_VertexBuffer as a sequence of triangles, a triangle fan, triangle strips, or a polyline. The indices can either specify a range of vertices (renders vertices in order), or a set of indices (renders vertices in specified order). The vertices specified by the indices are rendered as a single primitive chain using the primitive type. For each primitive type, the resulting geometry is as follows:

- A661_VERTEX_RENDER_LINES – the rendered geometry will be equivalent to a sequence of LINE commands starting at each even index
- A661_VERTEX_RENDER_LINE_LOOP – the rendered geometry will be equivalent to a single POLYLINE command from the first indexed vertex to the last and continuing to the first.
- A661_VERTEX_RENDER_POLYLINE – the rendered geometry will be equivalent to a single POLYLINE command from the first indexed vertex to the last.
- A661_VERTEX_RENDER_TRIANGLE – Each 3 indexed vertices define an independent TRIANGLE
- A661_VERTEX_RENDER_TRIANGLE_FAN – Indexed version of the TRIANGLE_FAN
- A661_VERTEX_RENDER_TRIANGLE_STRIP – Indexed version of the TRIANGLE_STRIP

The RenderSequential flag is used to govern whether this map item specifies a range of sequential vertices or a sequence of indexed vertices. If the range is sequential, parameters I1 and I2 specify a range, and no Vertex_Render_Segments are expected following this map item.

Table 3.3.22.2.1.26 – Vertex_Render_Start

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_VERTEX_RENDER_START
PrimitiveType	uchar	8	See description
VertexBufferIndex	ushort	16	Index of the MapHorz_VertexBuffer widget containing the vertex data
RenderSequential	uchar	8	Whether to render sequentially or as indices. A661_TRUE or A661_FALSE, If true, only the first 2 indices define the range
NumIndices	uchar	8	Number of indices in the expected index list. If <= 6, the command is a complete render
I1	ushort	16	First index number or range start
I2	ushort	16	Second index number or range end
I3	ushort	16	Third index if indexed
I4	ushort	16	Fourth index if indexed
I5	ushort	16	Fifth index if indexed
I6	ushort	16	Sixth index if indexed

3.3.22.2.1.27 Vertex_Render_Segment

The Vertex_Render_Segment adds a segment to a vertex render start map item defined with the RenderSequential flag set to A661_FALSE;

Table 3.3.22.2.1.27 – Vertex_Render_Segment

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_VERTEX_RENDER_SEGMENT
UnusedPad	N/A	8	0
Indices	ushort x 8	8*16	8 indices

3.3.22.2.1.28 Vertex_Render_End

The Vertex_Render_End terminates vertex render map items defined with the RenderSequential flag set to A661_FALSE;

Table 3.3.22.2.1.28 – Vertex_Render_End

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_VERTEX_RENDER_END
NumIndices	uchar	8	Number of indices used in this list
Indices	ushort x 8	8*16	Unused indices set to 0

3.3.22.2.1.29 Symbol_Parkable and Symbol_Parkable_Interactive

The Symbol_Parkable and Symbol_Parkable_Interactive map items allow defining a symbol that is shown on the map based on a Map Boundary (see widget defined in Section 3.9.15).

Table 3.3.22.2.1.29 – Symbol_Parkable and Symbol_Parkable_Interactive

Name	Type	Size (bits)	Description
ItemIndex	ushort	16	Position of the item in the MapItem_List
ItemType	uchar	8	A661_SYMBOL_PARKABLE, A661_SYMBOL_PARKABLE_INTERACTIVE
EndFlag	uchar	8	A661_TRUE, A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_WAYPOINT
ParkedSymbolType	ushort	16	EXAMPLES: SYMBOL_AIRPORT
RelativePosition	uchar	8	A661_FALSE, A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
Reserved	uchar	8	
UnusedPad	N/A	16	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.

3.3.22.2.1.30 Symbol_Rotated_Parkable and Symbol_Rotated_Parkable_Interactive

The Symbol_Parkable_Rotated and Symbol_Parkable_Rotated_Interactive map items allow defining a symbol that is shown on the map based on a Map Boundary (see widget defined in Section 3.9.15).

- If this Symbol is inside the MapBoundary: It is drawn exactly as if it was a SymbolRotated.
- If this Symbol is outside the MapBoundary: Apart from the fact that it will use the ParkedSymbolType to be drawn, how the Symbol is drawn is implementation dependent. For example, an implementation could not rotate the Symbol if it is outside the MapBoundary.

Table 3.3.22.2.1. 30 – Symbol_Rotated_Parkable and Symbol_Rotated_Parkable_Interactive

Name	Type	Size (bits)	Description
ItemIndex	ushort	16	Position of the item in the MapItem_List
ItemType	uchar	8	A661_SYMBOL_ROTATED_PARKABLE, A661_SYMBOL_ROTATED_PARKABLE_INTERACTIVE
EndFlag	uchar	8	A661_TRUE, A661_FALSE
SymbolType	ushort	16	EXAMPLES: SYMBOL_HOLD_LEFT
ParkedSymbolType	ushort	16	EXAMPLES: SYMBOL_HOLD_LEFT_PARKED
RelativePosition	uchar	8	A661_FALSE, A661_TRUE When RelativePosition is true, then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
Reserved	uchar	8	
UnusedPad	N/A	16	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Orientation	fr(180)	32	Orientation of Symbol (clockwise is positive rotation) relative to True North.

3.3.22.2.1.31 Symbol_Target_Parkable and Symbol_Target_Parkable_Interactive

The Symbol_Target_Parkable and Symbol_Target_Parkable_Interactive map items allow defining a symbol that is shown on the map based on a Map Boundary (see widget defined in Section 3.9.15).

- If this Symbol is inside the MapBoundary: It is drawn exactly as if it was a SymbolTarget.
- If this Symbol is outside the MapBoundary: Apart from the fact that it will use the ParkedSymbolType to be drawn, how the Symbol is drawn is implementation dependent. For example, an implementation could not rotate the Symbol if it is outside the MapBoundary.

Table 3.3.22.2.1.31 – Symbol_Target_Parkable and Symbol_Target_Parkable_Interactive

Name	Type	Size (bits)	Description
ItemIndex	ushort	16	Position of the item in the MapItem_List
ItemType	uchar	8	A661_SYMBOL_TARGET_PARKABLE, A661_SYMBOL_TARGET_PARKABLE_INTERACTIVE
EndFlag	uchar	8	A661_TRUE, A661_FALSE
SymbolType	ushort	16	EXAMPLES: AIR_FRIEND_TRACK
ParkedSymbolType	ushort	16	EXAMPLES: AIR_FRIEND_TRACK_PARKED
RelativePosition	uchar	8	A661_FALSE, A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
Reserved	uchar	8	
Length	ushort	16	Velocity/Distance/Altitude illustrated by variable length part of the symbol (Knots/Nautical Miles/Feet). The choice of units is an OEM decision. The units for the MIL instance can be determined by the value of the SymbolType parameter, or in some other way.
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Y	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b.
Orientation	fr(180)	32	Orientation of Symbol relative to True North

3.3.22.2.1.32 Parking_Line_Start

The Parking_Line_Start allows the definition of the start of a line which is shown on the map based on a Map Boundary (see widget defined in Section 3.9.15).

Table 3.3.22.2.1.32 – Parking_Line_Start

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	Position of the item in the MapItem_List
Item Type	uchar	8	A661_PARKING_LINE_START
UnusedPad	NA	8	Unused Pad
X1	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y1	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Orientation	fr(180)	32	Orientation of Symbol relative to True North
SymbolType	ushort	16	Unique identifier of the symbol to be drawn at the first vertex
ParkedSymbolType	ushort	16	Unique identifier of the symbol to be drawn at the first vertex when parked

3.3.22.2.1.33 Parking_Line_End

The Parking_Line_End allows the definition of the end of a line which is shown on the map based on a Map Boundary (see widget defined in Section 3.9.15).

Table 3.3.22.2.1.33 – Parking_Line_End

Parameter	Type	Size (bits)	Description
Item Index	ushort	16	Position of the item in the MapItem_List
Item Type	uchar	8	A661_PARKING_LINE_END
UnusedPad	NA	8	Unused Pad
X2	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y2	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
SymbolType	ushort	16	Unique identifier of the symbol to be drawn at the second vertex
ParkedSymbolType	ushort	16	Unique identifier of the symbol to be drawn at the second vertex when parked

3.3.22.2.1.34 Boundary_Check

A Boundary_Check item indicates that boundary checking should be performed on subsequent map items for types to which boundary checking applies (i.e., symbol-based types).

A Boundary_Check item with an ApplyCheck value of A661_TRUE defines the start of a set of map items for which boundary checks are performed on applicable types, and a Boundary_Check item with an ApplyCheck value of A661_FALSE defines the end of that set.

Boundary checks are not performed on any map items preceding the first Boundary_Check item with ApplyCheck set to A661_TRUE. If no item with ApplyCheck set to A661_FALSE is provided after one which was set to A661_TRUE, then all subsequent map items of eligible type are boundary checked.

Boundary_Check is defined in Table 3.3.22.2.1.32.

Table 3.3.22.2.1.34 – Boundary_Check

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_BOUNDARY_CHECK
ApplyCheck	uchar	8	A661_FALSE A661_TRUE

3.3.22.2.1.35 Legend_Readout_Format_String

Similar to the ItemStyle MapItem, for formatting the Legend_Readout the CDS must apply the last defined Legend_Readout_Format_String found in the Map Item List. If no A661_LEGEND_READOUT_FORMAT_STRING has been provided, the CDS will apply a default Legend_Readout_Format_String. The way the Readout_Format of this MapItem work is specified in Section 3.2.6, "Formatted Numeric Values."

Note that this MapItem does not prevent the use of an ItemStyle on the same Legend_Readouts. For example, a Legend_Readout can have both a

ReadoutFormat (to format its numeric value) and an ItemStyle (to apply a style to the corresponding Legend_Readouts).

Table 3.3.22.2.1.35 – Legend_Readout_Format_String

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT_FORMAT_STRING
EndFlag	uchar	8	A661_TRUE A661_FALSE
ReadoutFormat	{uchar}+ (not 'string' because it might not be null terminated)	{32}+	Max 16 characters. Section 3.2.9.5 defines the proper string termination and padding.

3.3.22.2.1.36 Legend_Readout

The MapItem Legend_Readout simplifies the presentation of numbers (number or index of Waypoints, Runway heading, etc.) use cases in the point of view of the UA.

Table 3.3.22.2.1.36 – Legend_Readout

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT
EndFlag	uchar	8	A661_TRUE A661_FALSE
ReadoutValue	float	32	

The formatting of this MapItem apply the ReadoutFormat defined in the last Legend_Readout_Format_String MapItem.

This MapItem can be mixed with Legend MapItems to be able to present values with both formatted numbers and text.

3.3.22.2.1.37 Legend_Readout_Combo

Legend_Readout_Combo allows a Legend to be defined in a single MapItem as any combination of:

- LEGEND_READOUT,
- Popped-up LEGEND_READOUT (This Item is a basic LEGEND_READOUT, but it will appear only when the crew member selects the associated SYMBOL_xxx Item. Disappearance of the popup is implementation dependent),
- Highlighted LEGEND_READOUT (This Item is a basic LEGEND_READOUT, but it will appear only when the associated interactive SYMBOL_x Item is highlighted),

This allows fewer maps items to be used to define the same behavior. This MapItem works similar to the Legend_Combo MapItem.

One Legend_Readout_Combo map item with LegendTypes is set to:

Table 3.3.22.2.1.37-1 – Legend_Readout_Combo

LegendType	Equivalent to
A661_LEGEND_ONLY	One A661_LEGEND_READOUT item
A661_POPUP_ONLY	One popped-up A661_LEGEND_READOUT item
A661_HIGHLIGHT_ONLY	One highlighted A661_LEGEND_READOUT item
A661_LEGEND_AND_POPUP	One A661_LEGEND_READOUT item One popped-up A661_LEGEND_READOUT item With the same value
A661_LEGEND_AND_HIGHLIGHT	One A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value
A661_POPUP_AND_HIGHLIGHT	One popped-up A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value
A661_LEGEND_AND_POPUP_AND_HIGHLIGHT	One A661_LEGEND_READOUT item One popped-up A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value

Legend_Readout_Combo is defined in Table 3.3.22.2.1.25.

Table 3.3.22.2.1.37-2 – Legend_Readout_Combo

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT_COMBO
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendTypes	uchar	8	A661_LEGEND_ONLY A661_POPUP_ONLY A661_HIGHLIGHT_ONLY A661_LEGEND_AND_POPUP A661_LEGEND_AND_HIGHLIGHT A661_POPUP_AND_HIGHLIGHT A661_LEGEND_AND_POPUP_AND_HIGHLIGHT Value of 0 is reserved
UnusedPad	N/A	24	
ReadoutValue	float	32	

3.3.22.2.2 A661_ParameterStructure_BufferOfItems

A661_ParameterStructure_BufferOfItems as used for MapHorz_ItemList is defined in Table 3.3.22.2.2.

Table 3.3.22.2.2 – A661_ParameterStructure_BufferOfItems

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

3.3.22.2.3 A661_ParameterStructure_BlockedBufferOfItems

The BlockedBufferOfItems parameter only addresses update synchronization at the MapHorz_ItemList widget level. The collection of blocks is treated as a single atomic set. For example, if a buffer comprised of two blocks contains map items 1 to 200 in block 1, and 201 to 300 in block 2, all 300 map items will be drawn at the same time.

Update synchronization across multiple map item list widgets, or across multiple layers containing map widgets, is not addressed by this parameter.

A661_ParameterStructure_BlockedBufferOfItems as used for MapHorz_ItemList is defined in Table 3.3.22.2.3.

Table 3.3.22.2.3 – A661_ParameterStructure_BlockedBufferOfItems

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_BLOCKED_BUFFER_OF_MAPITEM
ClearFlag	1	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
NumberofItems	15	Number of Items modified by the command
LastBlock	8	Indicates the end of the blocked buffer command. A661_FALSE A661_TRUE
UpdateNumber	8	Unique number to indicate the sequence to which this block belongs.
BlockNumber	8	Number indicating the block's position in the sequence.
UnusedPad	8	
{ItemStructures}+	{32}+	

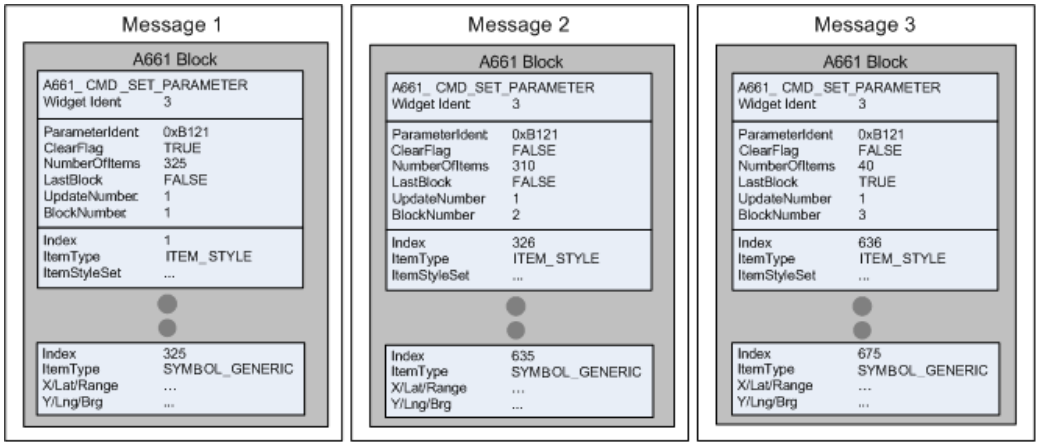


Figure 3.3.22.2.3 – Blocked_Buffer Parameter Structure

3.3.22.3 MapHorz_ItemList Interactive Map Items

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

Overlapping Active Areas:

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

Recommended Behavior For Segment Highlighting and Sending of Events:

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

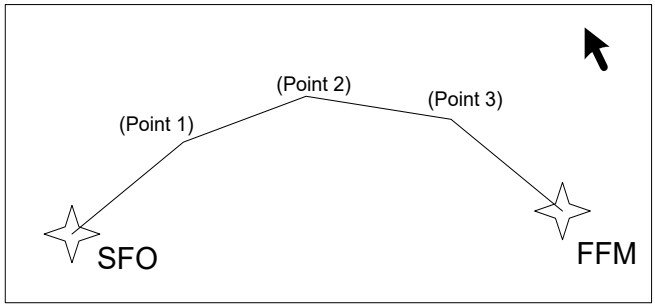


Figure 3.3.22.3-1 – A Line Sequence With Four Straight Line Segments

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

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

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

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

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

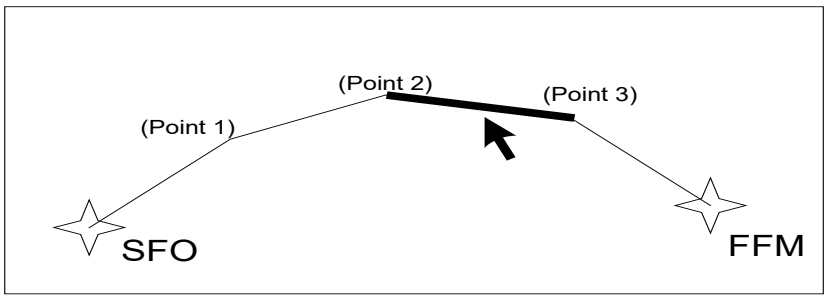


Figure 3.3.22.3-2 – Highlighting and Event Reporting Based on Individual Line Segments

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

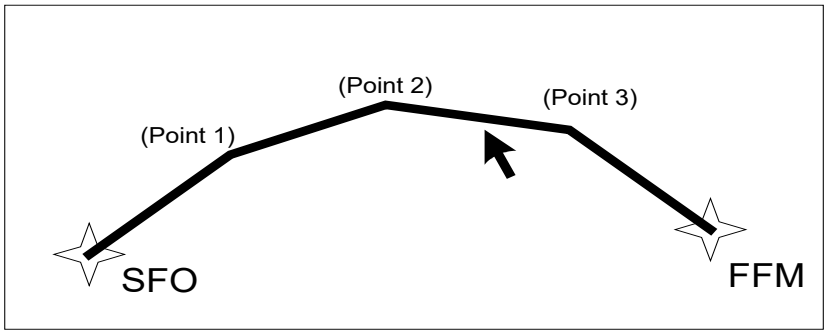


Figure 3.3.22.3-3 – The Entire Segment is Highlighted

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

1. To send the item index of the individual segment that was highlighted at the time of the click, or

2. To send the item index of the LINE_START_INTERACTIVE item (i.e., the same item regardless of which of its line segments was clicked on).

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

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

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

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

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

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

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

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

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

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

Note: Straight lines were used in the examples. The concept applies to arcs (LINE_ARC and LINE_ARC_INTERACTIVE) in the same way. The concept also applies to triangle strips and triangle fans regarding the highlighting and event reporting of individual triangles.

3.3.23 MapLegacy (DELETED)

This widget was deprecated by Supplement 4.This widget was deleted by Supplement 5.

3.3.24 MapHorz_Source

Categories:

- Map management
- Container
- Interactive

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

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

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

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

MapHorz_Source Parameters are defined in Table 3.3.24-1.

Table 3.3.24-1 – MapHorz_Source Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_SOURCE
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
Specific Parameters		
MapDataFormat	D	Format of the data contained by this MapHorz_Source. This parameter defines the coordinate system as well as the kind of transformation to apply on dynamic widgets contained by the MapHorz_Source. See values in the table below.
EventFlag	DR	**Type of Event to return:** • **A661_EVENT_NONE: no events will be sent** • **A661_VALIDATION: A661_EVT_SELECTION_MAP events will be sent** • **A661_VALIDATION_AND_WHEEL: A661_EVT_SELECTION_MAP and A661_EVT_INCREMENT events will be sent** • **A661_WHEEL: A661_EVT_INCREMENT events will be sent**
Coordinate Reference Point Latitude	R	This point is used by the CDS to know what reference should be used to compute the latitude/longitude positions of coordinates using the A661_MDF_DIST_DIST_FIX data format, in which the coordinates are relative to a reference point other than the PRP or Aircraft position. The Coordinate Reference Point (CRP) should not be used to rotate or translate map data; this should be done using the Projection Reference Point or Aircraft position. The CRP allows a means for a UA to provide map data relative to a static location independent of the display format or aircraft, such as coordinates from databases with fixed-point references (for example, airport map). Commentary: For instance, an airport map database may have coordinates relative to an aerodrome reference point (survey point). In a PLAN mode ND, the PRP may be a waypoint in the flight plan. The Coordinate Reference Point can be set to the latitude/longitude position of the aerodrome reference point. The CDS can then determine the latitude/longitude positions for distance/distance coordinates based on the Coordinate Reference Point, and project these positions based on the Projection Reference Point.
Coordinate Reference Point Longitude	R	

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

Table 3.3.24-2a – MapHorz_Source Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
MapDataFormat	uchar	8	A661_MDF_BRG_DIST_ACHDG A661_MDF_DIST_DIST A661_MDF_LAT_LONG A661_MDF_DIST_DIST_FIX A661_MDF_LEGACY
EventFlag	uchar	8	A661_EVENT_NONE A661_VALIDATION A661_VALIDATION_AND_WHEEL A661_WHEEL
UnusedPad	N/A	16	0

MapDataFormat values are defined in Table 3.3.24-2b.

Based on the MapDataFormat setting, the X and Y parameters for MapHorz_ItemList items can represent real world quantities such as Latitude, Range, Longitude, and Bearing. Units are as given in the following table.

Note that when Relative Positioning is being used (for MIL items having the RelativePosition parameter), the X and Y parameters are expressed in screen coordinates, and are encoded using data type “long” (32 bits signed), regardless of the MapDataFormat setting.

MapDataFormat Structure is defined in Table 3.3.24-2b.

Table 3.3.24-2b – MapDataFormat Values

Value	Alignment of +Y Axis	Origin	Positive Orientation	Units of Measure	Type (Note 1)
A661_MDF_BRG_DIST_ACHDG	X: Distance Y: Bearing	aircraft lat/long defined in MapHorz	X: specified by Y Y: clockwise from aircraft orientation defined in MapHorz	X: nautical miles Y: degrees	X: fr(32768) Y: fr(180)
A661_MDF_DIST_DIST (Note 2)	X: Distance Y: Distance	aircraft lat/long defined in MapHorz	X: east Y: north	X and Y: nautical miles	X: fr(32768) Y: fr(32768)
A661_MDF_DIST_DIST_FIX (Note 2)	X: Distance Y: Distance	CRP lat/long	X: east Y: north	X and Y: nautical miles	X: fr(32768) Y: fr(32768)
A661_MDF_LAT_LONG	X: Latitude Y: Longitude	prime meridian at the equator	X: north Y: east	X and Y: degrees	X: fr(180) Y: fr(180)
A661_MDF_LEGACY	various	various	N/A	various	various

Notes:

1. Type refers to the data type of the X and Y parameters in the descendants of the MapHorz_Source.
2. A661_MDF_DIST_DIST and A661_MDF_DIST_DIST_FIX use a Cartesian coordinate system where the Y-direction is parallel to the true north vector starting from the projection reference point. A661_MDF_DIST_DIST and A661_MDF_DIST_DIST_FIX are intended to measure short distances for which correlation errors related to the curvature of the earth are not significant; for example, with airport charts.

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	Dependent on MapDataFormat	32	See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b

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

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_INCREMENT
UnusedPad	N/A	16	0
NbOfIncrements	long	32	Number of input device increments. Interpretation of this value is implementation dependent.

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

Table 3.3.24-4b – MapHorz_Source Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
EventFlag	uchar	8	A661_EVENT_FLAG
CoordinateReferencePointLatitude	fr(180)	32	A661_CRP_LAT
CoordinateReferencePointLongitude	fr(180)	32	A661_CRP_LONG
CoordinateReferencePointLatitude, CoordinateReferencePointLongitude	fr(180) x 2	32 x 2	A661_CRP_LAT_LONG

3.3.25 MapHorz

Categories:

- Container
- Map management

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

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

Restriction:
None

MapHorz Parameters are defined in Table 3.3.25-1.

Table 3.3.25-1 – MapHorz Parameters

Parameters	Change	Description
WidgetType	D	A661_MAPHORZ
WidgetIdent	D	Unique identifier of the widget
ParentIdentifier	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
PosX	DR	The X position of the widget reference point (screen coordinate system)
PosY	DR	The Y position of the widget reference point (screen coordinate system)
SizeX	DR	Area size X
SizeY	DR	Area size Y
MapSynchronizationNumber	R	See Section 3.2.8.4
Reference coordinate system		
Projection Reference Point Latitude	R	This point is used by the CDS to know what reference should be used to run the projection algorithm. The CDS converts dynamic widget coordinate data into the MapHorz coordinate system. The MapHorz coordinate system is defined by: Reference point: PRP with lat/long coordinate Reference direction: True North Commentary: For the ND, PRP is the aircraft position for ARC and ROSE mode. For PLAN mode the PRP is a waypoint of the FPLN. In mode PLAN, the PRP can be populated by the FMS through the ND.
Projection Reference Point Longitude	R	
Equivalence between "MapHorz coordinate system" and "MapHorz Screen Coordinate system"		
ScreenReferencePointX	DR	X and Y Position of the PRP on the screen. This position is expressed in MapHorz Screen Coordinate System relative to the widget reference point (PosX, PosY)
ScreenReferencePointY	DR	
Range	DR	Geo-referenced range expressed in nm
ScreenRange	DR	Distance in screen units (hundredth of mm) equivalent to range.
Orientation parameters for latitude/longitude and TCAS coordinate like systems		
Orientation	R	Angle between True North and the up-direction of the display at the PRP (see Reference coordinate system, positive direction: counter-clockwise). If PRP Latitude is at a pole, the up-direction of the display should be the meridian identified by PRP Longitude.
AircraftOrientation	R	Orientation of the aircraft relative to the True North (positive direction: clockwise)
AircraftLatitude	R	Latitude of the aircraft
AircraftLongitude	R	Longitude of the aircraft
ProjectionType	D	Indicate which kind of projection will be applied on lat/long coordinate of dynamic widget. For example: LAMBERT POLAR

MapHorz Creation is defined in Table 3.3.25-2a.

Table 3.3.25-2a – MapHorz Creation Structure

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

MapHorz Event Structures: the structure of A661_EVT_ITEM_SYNCHRONIZATION is defined in Table 3.3.25-2b. This event is initiated by the transmission of an Item_Synchronization in a MapHorz_ItemList. See the definition of the Item_Synchronization in the MapHorz_ItemList for more details.

Table 3.3.25-2b – MapHorz Event Structures: A661_EVT_ITEM_SYNCHRONIZATION

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

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

Table 3.3.25-3 – MapHorz Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
Projection Reference Point Latitude	fr(180)	32	A661_PRP_LAT
Projection Reference Point Latitude, Projection Reference Point Longitude	fr(180) x 2	32 x 2	A661_PRP_LAT_LONG
Projection Reference Point Longitude	fr(180)	32	A661_PRP_LONG
ScreenReferencePointX	long	32	A661_PRP_SCREEN_X
ScreenReferencePointX, ScreenReferencePointY	long x 2	32 x 2	A661_PRP_SCREEN_XY
ScreenReferencePointY	long	32	A661_PRP_SCREEN_Y
Range	fr(32768)	32	A661_RANGE
ScreenRange	ulong	32	A661_SCREEN_RANGE
Orientation	fr(180)	32	A661_ORIENTATION
AircraftLatitude	fr(180)	32	A661_AC_LAT
AircraftLongitude	fr(180)	32	A661_AC_LONG
AircraftLatitude, AircraftLongitude	fr(180) x 2	32 x 2	A661_AC_LAT_LONG
AircraftOrientation	fr(180)	32	A661_AC_ORIENTATION
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY

3.3.26 MaskContainer

Categories:

- Container

Description:
A MaskContainer widget applies a mask to a group of widgets to implement clipping **that is not limited to rectangular regions**. A mask can be represented by a picture or a symbol defining the clipping areas, but the exact usage is implementation dependent. A mask is 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	DR	X position of the mask. Note that this does not reposition widgets contained within the MaskContainer, only the mask itself.
PosY	DR	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.
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.
MotionAllowed	**D**	**Capability to change PosX, PosY at runtime.**

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	
MotionAllowed	**uchar**	**8**	**A661_FALSE** **A661_TRUE**
Unused pad	**N/A**	**8**	

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
Visible	uchar	8	A661_VISIBLE
MaskReference	ushort	16	A661_MASK_REFERENCE
MaskEnabled	uchar	8	A661_MASK_ENABLED
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**

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
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's descendants to be interactive
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
MotionAllowed	D	Capability to change PosX, PosY, SizeX, SizeY at run time
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget

Panel Creation Structure is defined in Table 3.3.27-2.

Table 3.3.27-2 – Panel Creation Structure

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

No event is associated with the Panel widget.

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

Table 3.3.27-3 – Panel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**

Parameters	Change	Description
Alignment	DR	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	DR	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

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

Table 3.3.28-2 – Picture PushButton Creation Structure

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

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

Table 3.3.28-3 – Picture PushButton Event Structures: A661_EVT_SELECTION

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

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

Table 3.3.28-4 – Picture PushButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
LabelString	string	{32}+	A661_STRING
PictureReference	ushort	16	A661_PICTURE_REFERENCE
StyleSet	ushort	16	A661_STYLE_SET
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**

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 are 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 interactive
ToggleState	DR	Inner state of the ToggleButton SELECTED UNSELECTED
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
AlternateFlag	DR	True: Enables switching between the two strings (and two pictures) according to the inner state. CDS will switch strings when the inner state changes. False: Disables switching, “AlternateLabelString” and AlternatePictureReference are not used. “LabelString” and “PictureReference” are used for both inner states.
LabelString	DR	Label of the PictureToggleButton Label used for UNSELECTED state
AlternateLabelString	DR	Label of the PictureToggleButton Label used for SELECTED state
PictureReference	DR	Picture on the PictureToggleButton Picture used for UNSELECTED state
AlternatePictureReference	DR	Picture on the PictureToggleButton Picture used for SELECTED state
Alignment	DR	Alignment of the text within the label area of the widget: LEFT RIGHT CENTER
PicturePosition	DR	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

PictureToggleButton Creation Structure is defined in Table 3.3.29-2.

Table 3.3.29-2 – PictureToggleButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE_TOGGLE_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
AlternateFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
PictureReference	ushort	16	
AlternatePictureReference	ushort	16	
PicturePosition	uchar	8	A661_LEFT A661_CENTER A661_RIGHT A661_TOP A661_BOTTOM
ToggleState	uchar	8	A661_UNSELECTED A661_SELECTED
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UnusedPad	N/A	8	0
LabelString	string	8 * string1 length	The string terminating NULL is used as string separator.
AlternateLabelString	string	8 * string2 length + Pad	Followed by zero, one, two, or three extra NULL for alignment of 32 bits.

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

Table 3.3.29-3 – PictureToggleButton Event Structures: A661_EVT_STATE_CHANGE

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

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

Table 3.3.29-4 – PictureToggleButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
ToggleState	uchar	8	A661_INNER_STATE_TOGGLE
StyleSet	ushort	16	A661_STYLE_SET
PictureReference	ushort	16	A661_PICTURE_REFERENCE
AlternatePictureReference	ushort	16	A661_ALTERN_PICTURE_REFERENCE
LabelString	string	{32}+	A661_STRING
AlternateLabelString	string	{32}+	A661_STRING_ALTERNATE
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**
AlternateFlag	**uchar**	**8**	**A661_ALTERNATE_FLAG**

3.3.30 PopUpPanel

Categories:

- Container
- Graphical representation
- Interactive

Description:
The PopUpPanel is a container intended for temporary display on top of all layers in its window. PopUpPanel has no static part.

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

PopUpPanel widget has clipping capability.

Restriction:

PopUpPanel Parameters are defined in Table 3.3.30-1.

Table 3.3.30-1 – PopUpPanel Parameters

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

PopUpPanel Creation Structure is defined in Table 3.3.30-2.

Table 3.3.30-2 – PopUpPanel Creation Structure

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

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

Table 3.3.30-3 – PopUpPanel Event Structures: A661_EVT_POPUP_PANEL_CLOSED

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

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

Table 3.3.30-4 – PopUpPanel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
StyleSet	ushort	16	A661_STYLE_SET
UAPositionFlag	uchar	8	A661_UA_POSITION_FLAG
AutomaticClosure	uchar	8	A661_AUTO_CLOSURE

3.3.31 PopUpMenu

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The PopUpMenu is a menu intended for temporary display on top of all layers in its window. PopUpMenu has no static part. PopUpMenu displays a list of selectable string entries, each of which represents either a control selection or sub-menu item. Crew interaction with a menu entry can cause another PopUpMenu to appear.

PopUpMenu visibility should be managed by the CDS using **implementation dependent** logic.

The OpeningMode parameter allows the UA some control over the position of the PopUpMenu. However, the CDS is responsible for defining the size and position of the PopUpMenu, as part of the implementation dependent look and feel.

Restriction:
None

PopUpMenu Parameters are defined in Table 3.3.31-1.

Table 3.3.31-1 – PopUpMenu Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_POP_UP_MENU
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	R	Visibility of the widget: A661_FALSE A661_TRUE Note: Widget is not visible at creation time.
StyleSet	DR	Referenced to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
OpeningMode	DR	OPEN_UP: the position (X,Y) is the bottom/left corner of the PopUpMenu OPEN_DOWN: the position (X,Y) is the top/left corner of the PopUpMenu CDS_DEPENDENT: Position defined by CDS using CCD Click location
NumberOfEntries	D	Number of entries in the PopUpMenu. It also includes the graphical separators.
MaxStringLength	D	**Maximum size in bytes (including the NULL terminator) of each string array entry.**
StringArray	DR	Array of Strings. Each String is the text label for a menu entry. Strings composed of only one NULL character will be interpreted as a graphical separator. The NULL string at the end of the array will not be interpreted.
PictureArray	DR	Array of Pictures, each of which is the Picture icon for a menu entry. If value is NULL, no picture is attached to the corresponding entry.
PopUpIdentArray	D	Array of PopUpMenu Widget Idents, each of which is the PopUpMenu that will be displayed when the crew interacts with the corresponding menu entry. WidgetIdent can only refer to another PopUpMenu. If WidgetIdent is NULL, no PopUpMenu is attached to the menu entry.
EnableArray	DR	Ability for each PopUpMenu entry: Refer to Table 4.5.4.5.7

PopUpMenu Creation Structure is defined in Table 3.3.31-2.

Table 3.3.31-2 – PopUpMenu Creation Structure

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

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

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

Table 3.3.31-3 – PopUpMenu Event Structures: A661_EVT_POPUP_CLOSED

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

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

Table 3.3.31-4 – PopUpMenu Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
StyleSet	ushort	16	A661_STYLE_SET
StringArray [up to NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY
EnableArray (Deprecated by Supplement 4)	N/A	16	A661_ENABLE_ARRAY
EnableArray [up to NumberOfEntries]	N/A	{32}+	A661_ENABLE_ENTRY_ARRAY
PictureArray (Deprecated by Supplement 4)	N/A	24	A661_PICTURE_ARRAY Refer to definition in Table 3.3.31.1-1 below.
PictureArray [up to NumberOfEntries]	N/A	{32}+	A661_PICTURE_ENTRY_ARRAY
StringArray [up to NumberOfEntries] and EnableArray [up to NumberOfEntries] and PictureArray [up to NumberOfEntries]	N/A	{32}+	A661_ENTRY_POP_UP_ARRAY
OpeningMode	uchar	8	A661_OPENING_MODE

3.3.31.1 PopUp Specific A661_ParameterStructure

The structure described in Table 3.3.31.1-1 was deprecated by Supplement 4 and replaced by A661_ParameterStructure_2ByteArray. Refer to Table 4.5.4.5.13.

A661_ParameterStructure_PictureArray is defined in Table 3.3.31.1-1.

Table 3.3.31.1-1 – A661_ParameterStructure_PictureArray

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

3.3.32 PopUpMenuButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The PopUpMenuButton widget has a static part (a Button) which, when selected by a crew member, causes the display of the popup part of the PopUpMenuButton. The popup part of the PopUpMenuButton is similar to a PopUpMenu widget.

The UA is responsible for defining the position of the popup part of this widget. The OpeningMode parameter can be used to further control the position of the popup part relative to the static part.

The method for closing the popup part of the widget is implementation dependent.

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	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	DR	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
Specific Parameters of PopUp		
PopUpPosX	DR	The X position of the popup part reference point
PopUpPosY	DR	The Y position of the popup part reference point
PopUpSizeX	DR	The X dimension size (width) of the popup part
PopUpSizeY	DR	The Y dimension size (height) of the popup part
OpeningMode	DR	OPEN_UP: the position (PopUpPosX/Y) is the bottom/left point corner of the popup part OPEN_DOWN: the position (PopUpPosX/Y) is the top/left point corner of the popup part
NumberOfEntries	D	Number of entries in the popup part. It also includes the graphical separators.

Parameters	Change	Description
MaxStringLengthPopUp	D	**Maximum size in bytes (including the NULL terminator) of each string array entry.**
StringArray	DR	Array of Strings. Each String is the text label for a menu entry. Strings composed of only one NULL character will be interpreted as a graphical separator. The NULL string at the end of the array will not be interpreted.
PictureArray	DR	Array of Pictures, each of which is the Picture icon for a menu entry. If value is NULL, no picture is attached to the corresponding entry.
PopUpIdent Array	D	Array of PopUpMenu Widget Idents, each of which is the PopUpMenu that will be displayed when the crew interacts with the corresponding menu entry. WidgetIdent can only refer to another PopUpMenu. If WidgetIdent is NULL, no PopUpMenu is attached to the menu entry.
EnableArray	DR	Ability for each entry in the popup part: Refer to Table 4.5.4.5.7
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

PopUpMenuButton Creation Structure is defined in Table 3.3.32-2.

Table 3.3.32-2 – PopUpMenuButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_POP_UP_MENU_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
PopupPosX	long	32	
PopupPosY	long	32	
PopupSizeX	ulong	32	
PopupSizeY	ulong	32	
MaxStringLength	ushort	16	
MaxStringLengthPopUp	ushort	16	
PictureReference	ushort	16	
NumberOfEntries	uchar	8	
PicturePosition	uchar	8	A661_CENTER A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM

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

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

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

Table 3.3.32-3 – PopUpMenuButton Event Structures: A661_EVT_POPUP_CLOSED

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

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

Table 3.3.32-4 – PopUpMenuButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
LabelString	string	{32}+	A661_STRING
PictureReference	ushort	16	A661_PICTURE_REFERENCE
StringArray [up to NumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY
EnableArray (Deprecated by Supplement 4)	N/A	16	A661_ENABLE_ARRAY
EnableArray [up to NumberOfEntries]	N/A	{32}+	A661_ENABLE_ENTRY_ARRAY
PictureArray (Deprecated by Supplement 4)	N/A	24	A661_PICTURE_ARRAY Refer to definition in Table 3.3.31.1-1
PictureArray [up to NumberOfEntries]	N/A	{32}+	A661_PICTURE_ENTRY_ARRAY
StringArray [up to NumberOfEntries] and EnableArray [up to NumberOfEntries] and PictureArray [up to NumberOfEntries]	N/A	{32}+	A661_ENTRY_POP_UP_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
OpeningMode	**uchar**	**8**	**A661_OPENING_MODE**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**
PopupPosX	**long**	**32**	**A661_POPUP_POS_X**
PopupPosY	**long**	**32**	**A661_POPUP_POS_Y**
PopupSizeX	**ulong**	**32**	**A661_POPUP_SIZE_X**
PopupSizeY	**ulong**	**32**	**A661_POPUP_SIZE_Y**

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
Alignment	DR	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelString	DR	String of the PushButton
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

PushButton Creation Structure is defined in Table 3.3.33-2.

Table 3.3.33-2 – PushButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PUSH_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	

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

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

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

Table 3.3.33-3 – PushButton Event Structures: A661_EVT_SELECTION

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

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

Table 3.3.33-4 – PushButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
LabelString	string	{32}+	A661_STRING
StyleSet	ushort	16	A661_STYLE_SET
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.3.34 RadioBox

Categories:

- Container

Description:
A RadioBox widget manages the visibility and the interactivity of a group of Buttons (CheckButtons or ToggleButtons). It enables a crew member to select one Button out of “n” exclusive ones. At a given time one item maximum can be SELECTED. A selection of a selected item by a crew member is without effect. Nevertheless, the UA can deselect the selected item (through setParameter command) to create a

RadioBox without selection. The Buttons contained in the RadioBox should be individually defined with the RadioBox as a parent widget. RadioBox does not have any graphical representation.

Restriction:
The children of the RadioBox will be positioned relative to the parent of the RadioBox.

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

RadioBox Parameters are defined in Table 3.3.34-1.

Table 3.3.34-1 – RadioBox Parameters

Parameters	Change	Description
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's descendants to be interactive

RadioBox Creation Structure is defined in Table 3.3.34-2.

Table 3.3.34-2 – RadioBox Creation Structure

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

The RadioBox widget does not send any event.

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

Table 3.3.34-3 – RadioBox Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE

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 ancestor with a PosX, PosY reference point.

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

RotationContainerParameters are defined in Table 3.3.35-1.

Table 3.3.35-1 – RotationContainerParameters

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

RotationContainer Creation Structure is defined in Table 3.3.35-2.

Table 3.3.35-2 – RotationContainer Creation Structure

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

The RotationContainer widget does not send any event.

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

Table 3.3.35-3 – RotationContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
CenterX, CenterY	long x 2	32 x 2	A661_CENTER_XY
CenterX	long	32	A661_CENTER_X
CenterY	long	32	A661_CENTER_Y
RotationAngle	fr(180)	32	A661_ROTATION_ANGLE

3.3.36 ScrollPanel

Categories:

- Container
- Graphical representation
- Interactive

Description:
A scroll container is composed of two elements:

Frame, at fixed location. This region is defined by the parameters PosX, PosY, SizeX, SizeY.

Sheet, larger than the Frame, at a variable location with respect to the Frame. This region is defined by the variables FrameX, FrameY, SizeXsheet, SizeYsheet.

The sheet X/Y location is expressed in terms of the distance from the Sheet origin to the Frame origin, as shown in Figure 3.3.36.

The scrolling function is allowed by DeltaX, DeltaY parameters, which provide to the CDS the displacement of the sheet to apply when a crew member initiates an action with the scroll controls. The type of scroll controls provided are implementation dependent.

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

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

The children of the ScrollPanel widget are positioned relative to the origin of the sheet.

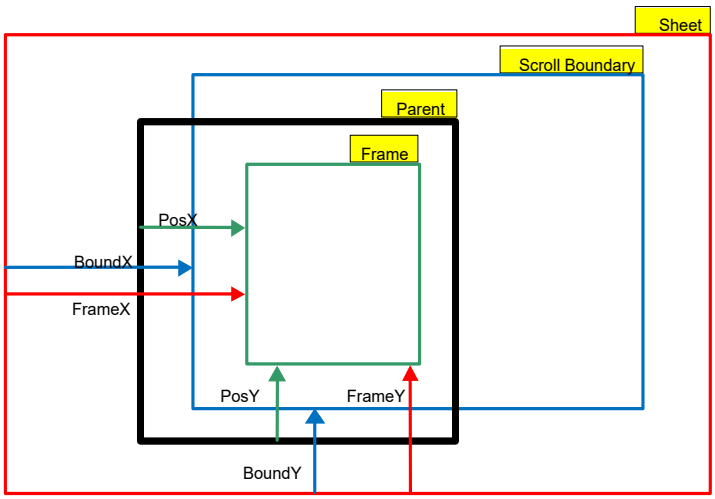


Figure 3.3.36 – Frame Standard Coordinate System

The direction of the arrows in the figure above indicates the sign of the parameter. In this case all the parameters are positive.

ScrollPanel Parameters are defined in Table 3.3.36-1.

Table 3.3.36-1 – ScrollPanel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SCROLL_PANEL
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Style Set	DR	Reference to predefined graphical characteristics inside CDS
Enable	DR	Ability of the widget to be interactive
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
LineDeltaX	DR	Increment/Decrement to apply to FrameX when line scroll controls are enabled.
LineDeltaY	DR	Increment/Decrement to apply to FrameY when line scroll controls are enabled.
PageDeltaX	DR	Increment/Decrement to apply to FrameX when page scroll controls are enabled.
PageDeltaY	DR	Increment/Decrement to apply to FrameY when page scroll controls are enabled.
HomeX	DR	A predefined X location for the sheet. The CDS may set FrameX to this value in certain implementation dependent conditions.
HomeY	DR	A predefined Y location for the sheet. The CDS may set FrameY to this value in certain implementation dependent conditions.
FrameX	DR	Specifies the sheet X location as the distance from the left edge of the sheet to the left edge of the frame.
FrameY	DR	Specifies the sheet Y location as the distance from the bottom edge of the sheet to the bottom edge of the frame.
SizeXsheet	DR	X dimension size of the sheet
SizeYsheet	DR	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 changes of the sheet location following crew member actions.
HorizontalScroll	DR	Absent/Top/Bottom/Left/Right
VerticalScroll	DR	Absent/Left/Right/Top/Bottom
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

ScrollPanel Creation Structure is defined in Table 3.3.36-2.

Table 3.3.36-2 – ScrollPanel Creation Structure

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

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

Table 3.3.36-3 – ScrollPanel Event Structures: A661_EVT_FRAME_POS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_FRAME_POS_CHANGE
UnusedPad	N/A	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
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
HomeX	**long**	**32**	**A661_HOME_X**
HomeY	**long**	**32**	**A661_HOME_Y**
FrameX	long	32	A661_FRAME_X
FrameY	long	32	A661_FRAME_Y
FrameX, FrameY	long x 2	32 x 2	A661_FRAME_XY
BoundX	long	32	A661_BOUND_X
BoundY	long	32	A661_BOUND_Y
BoundX, BoundY	long x 2	32 x 2	A661_BOUND_XY
SizeXbound	ulong	32	A661_BOUND_SIZE_X
SizeYbound	ulong	32	A661_BOUND_SIZE_Y
SizeXbound, SizeYbound	ulong x 2	32 x 2	A661_BOUND_SIZE_XY
EntryValidation	uchar	8	A661_ENTRY_VALID
LineDeltaX	**ulong**	**32**	**A661_LINE_DELTA_X**
LineDeltaY	**ulong**	**32**	**A661_LINE_DELTA_Y**
PageDeltaX	**ulong**	**32**	**A661_PAGE_DELTA_X**
PageDeltaY	**ulong**	**32**	**A661_PAGE_DELTA_Y**
SizeXsheet	**ulong**	**32**	**A661_SHEET_SIZE_X**
SizeYsheet	**ulong**	**32**	**A661_SHEET_SIZE_Y**
HorizontalScroll	**uchar**	**8**	**A661_HORIZONTAL_SCROLL**
VerticalScroll	**uchar**	**8**	**A661_VERTICAL_SCROLL**

3.3.37 ScrollList

Categories:

- Graphical Representation
- Interactive
- Text string

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

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

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

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

There must be two different numbering spaces for scroll list entries since there are two different lists and the UA managed list can be a larger size than the CDS managed list. For UA indexing, the range is 1 to NumberOfUAEntries and index 1 is the first item in the UA list. For CDS indexing, the range is 1 to NumberOfEntries and index 1 refers to the first item in the CDS managed list. All the widget parameters specifying indices are in terms of CDS indexing except for FirstAccessibleUAEntry. Since the CDS does not have a copy of the UA managed list, the FirstAccessibleUAEntry is only useful in order to provide some visual feedback about the relative position of the visible entries within the UA list.

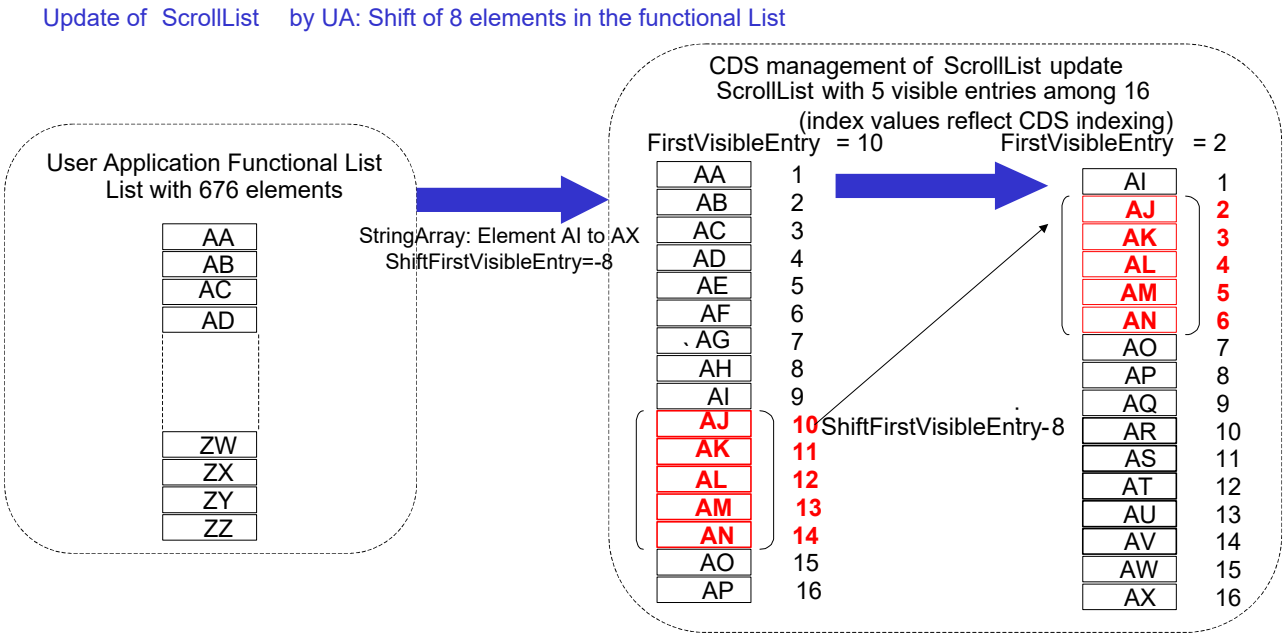


Figure 3.3.37-1 – Scroll List Update using ShiftFirstVisibleEntry and LabelStringArray

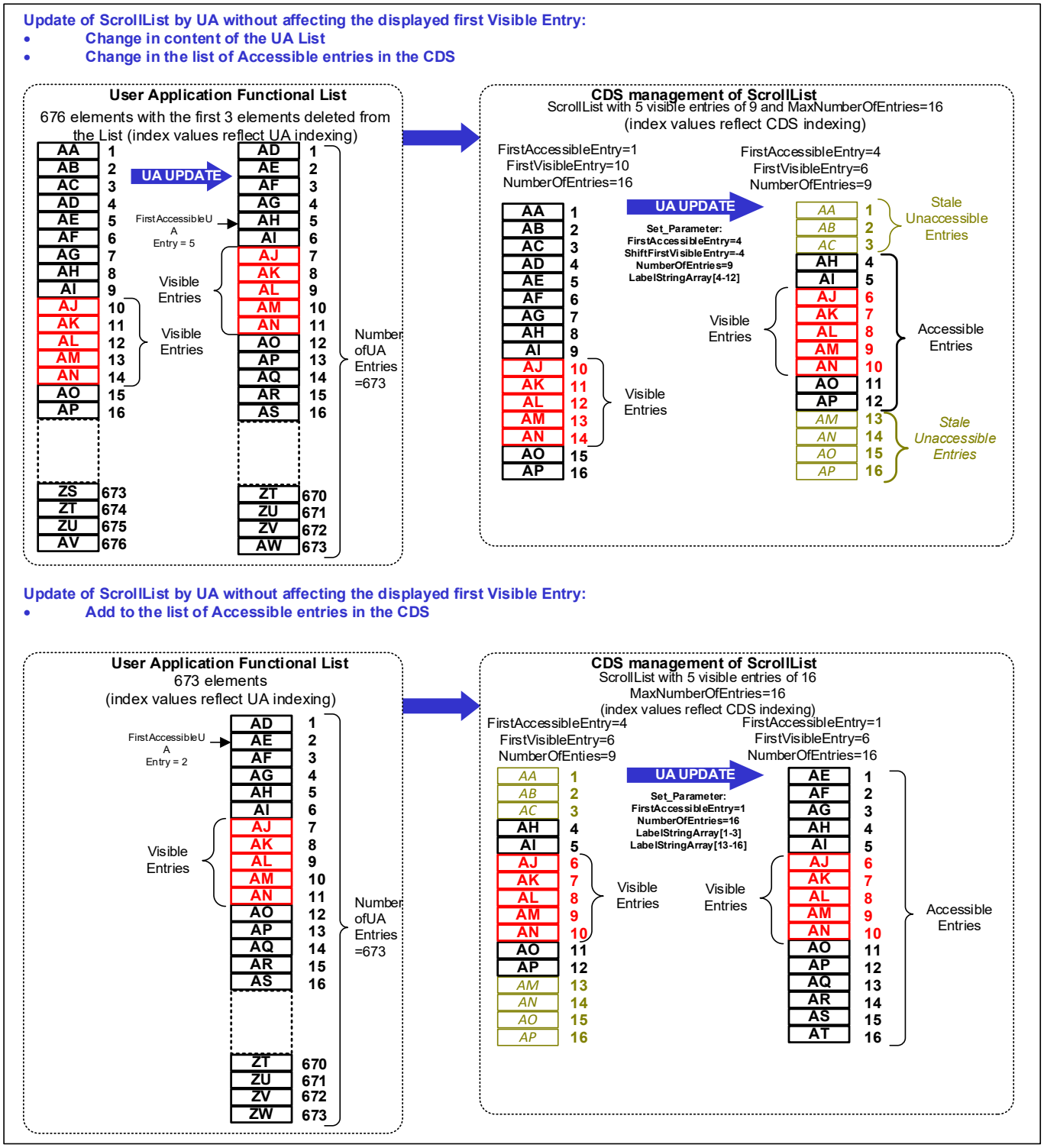


Figure 3.3.37-2 – Scroll List Update using ShiftFirstVisibleEntry and FirstAccessibleEntry

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

ScrollList Parameters are defined in Table 3.3.37-1.

Table 3.3.37-1 – ScrollList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SCROLL_LIST
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
NumberOfEntries	DR	Number of accessible entries
MaxNumberOfEntries	D	Max number of entries to be managed by the CDS
FirstVisibleEntry	DR	Index into LabelStringArray/EnableArray of the entry appearing as the first entry in the visible area of the ScrollList
ShiftFirstVisibleEntry	R	Index shift to be applied to the current FirstVisibleEntry value. That is, when a UA wants to add or delete entries above the FirstVisibleEntry, this parameter allows the UA to cause the CDS to change the value of FirstVisibleEntry by the number of deleted/inserted entries. FirstVisibleEntry is updated each time the ShiftFirstVisibleEntry parameter is received.
FirstAccessibleEntry	DR	The Entry number of the first entry in the LabelStringArray/EnableArray which the CDS manages as part of the scrollable items in the ScrollList.
NumberOfUAEntries	R	Number of entries in the UA managed list of Items.
FirstAccessibleUAEntry	R	Index into the UA managed list of items that corresponds to the FirstAccessibleEntry parameter. This parameter combined with NumberOfUAEntries allows the CDS to display the positioning of the visible entries in the ScrollList relative to the UA managed list. Note that not all CDS implementations display this information and may ignore these parameters.
FlagReportVisibleEntry	D	If True, CDS will report change on first visible entry following crew member actions.
SelectedEntry	DR	Currently selected Entry index

Parameters	Change	Description
DefaultStyleText	D	NULL character: Escape sequence not used, entries in the ScrollList are simple labels. “TOutline⊗TBackColor⊗TForeColor⊗TFont”
MaxStringLength	D	**Maximum size in bytes (including the NULL terminator) of each string array entry.**
Alignment	D**R**	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelStringArray	DR	Label array of the ScrollList
EnableArray	DR	Ability for each Entry on the ScrollList to be selected by a crew member: TRUE FALSE
VerticalScroll	D**R**	ABSENT LEFT RIGHT TOP BOTTOM
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

ScrollListCreation Structure is defined in Table 3.3.37-2.

Table 3.3.37-2 – ScrollList Creation Structure

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

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
VerticalScroll	uchar	8	A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT A661_ABSENT
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
FlagReportVisibleEntry	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
DefaultStyleText	uchar x 12	96	
EnableArray [NumberOfEntries]	{uchar}+	8 * NumberOfEntries	Enable status of 'NumberOfEntries' Entries from FirstAccessibleEntry
LabelStringArray [NumberOfEntries]	{string}+	8 * string lengths + Pad	There are "NumberOfEntries" strings. Each string terminating NULL is used as string separator. The complete string list is followed by zero, one, two, or three NULL character(s) to be 32 bits aligned.

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

Table 3.3.37-3 – ScrollList Event Structures: A661_EVT_SEL_ENTRY_CHANGE

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

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

Table 3.3.37-4 – ScrollList Event Structures: A661_EVT_FIRST_VIS_ENTRY_CHANGE

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

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

Table 3.3.37-5 – ScrollList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
NumberOfUAEntries	ushort	16	A661_NUMBER_OF_UA_ENTRIES
FirstAccessibleEntry	ushort	16	A661_FIRST_ACCESS_ENTRY
FirstAccessibleUAEntry	ushort	16	A661_FIRST_ACCESS_UA_ENTRY
FirstVisibleEntry	ushort	16	A661_FIRST_VISIBLE_ENTRY
ShiftFirstVisibleEntry	short	16	A661_SHIFT_FIRST_VISIBLE_ENTRY
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
LabelStringArray[up to MaxNumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY
EnableArray	N/A	16	A661_ENABLE_ARRAY Refer to Table 4.5.4.5.7 (Deprecated by Supplement 4)
EnableArray [up to Max NumberOfEntries]	N/A	{32}+	A661_ENABLE_ENTRY_ARRAY
LabelStringArray [up to MaxNumberOfEntries] and EnableArray [up to MaxNumberOfEntries]	**{string}+**	{32}+	A661_ENTRY_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
VerticalScroll	**uchar**	**8**	**A661_VERTICAL_SCROLL**

3.3.38 Symbol

Categories:

- Graphical Representation

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
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. **See Section 3.1.3.3.1.**
SymbolReference	DR	Reference of the symbol stored in the CDS

Symbol Creation Structure is defined in Table 3.3.38-2.

Table 3.3.38-2 – Symbol Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SYMBOL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
MotionAllowed	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
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
RotationAngle	fr(180)	32	A661_ORIENTATION
ColorIndex	uchar	8	A661_COLOR_INDEX
SymbolReference	ushort	16	A661_SYMBOL_REFERENCE

3.3.39 TabbedPanel

Categories:

- Container
- Graphical Representation
- Text string

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

The TabbedPanel widget is not interactive; however, it contains a NextFocusedWidget 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 interactive
StyleSet	DR	Reference to graphical characteristics defined inside CDS. The StyleSet will influence only the label or picture displayed on the button associated with the TabbedPanel.
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.

Parameters	Change	Description
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter.
Specific parameters		
LabelString	DR	Label of the tab
Alignment	DR	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
InsetSize	DR	Size of the button associated with the TabbedPanel, in the direction of the text writing, in screen units (millimeters).
Picture Reference	DR	Picture reference among available picture inside CDS
PicturePosition	DR	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM

COMMENTARY

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

TabbedPanel Creation Structures are defined in Table 3.3.39-2.

Table 3.3.39-2 – TabbedPanel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TABBED_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
PictureReference	ushort	16	
PicturePosition	uchar	8	A661_CENTER A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
UnusedPad	N/A	8	0
InsetSize	ulong	32	
LabelString	string	8 * string length + Pad	Followed by zero, one, two, or three extra NULL for alignment of 32 bits.

The TabbedPanel widget does not send any event.

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

Table 3.3.39-3 – TabbedPanel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
LabelString	string	{32}+	A661_STRING
PictureReference	ushort	16	A661_PICTURE_REFERENCE
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
InsetSize	**ulong**	**32**	**A661_INSET_SIZE**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**

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 TabbedPanels inside the TabbedPanelGroup widget occupy the same display space, and only one may be displayed at a time. The displayed TabbedPanel is the one referenced by the "ActiveTabbedPanelID" parameter.

Note: The effect of having a 0 value for the ActiveTabbedPanelID is implementation dependent.

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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
TabPosition	DR	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 buttons. FALSE: The button size is defined by the TabbedPanel parameter InsetSize. 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	Widget Ident of the active TabbedPanel or Connector
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

TabbedPanelGroup Creation Structure is defined in Table 3.3.40-2.

Table 3.3.40-2 – TabbedPanelGroup Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TABBED_PANEL_GROUP
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	

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

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

Table 3.3.40-3 – TabbedPanelGroup Event Structures: A661_EVT_TABBED_PANEL_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_TABBED_PANEL_CHANGE
ActiveTabbedPanelID	ushort	16	Widget Ident of the active TabbedPanel or Connector

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
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
ActiveTabbedPanelID	ushort	16	A661_ACTIVE_TABBED_PANEL
EntryValidation	uchar	8	A661_ENTRY_VALID
TabPosition	**uchar**	**8**	**A661_TAB_POSITION**

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 interactive
ToggleState	DR	Inner state of the ToggleButton UNSELECTED SELECTED
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
AlternateFlag	DR	True: Enables switching between the two strings according to the inner state. CDS will switch strings when the inner state changes. False: Disables switching, “AlternateLabelString” is not used. “LabelString” is used for both inner states.
Alignment	DR	Alignment of the text within the label area of the widget LEFT RIGHT CENTER
LabelString	DR	Label of the ToggleButton Label used for UNSELECTED state
AlternateLabelString	DR	Label of the ToggleButton Label used for SELECTED state
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

ToggleButton Creation Structure is defined in Table 3.3.41-2.

Table 3.3.41-2 – ToggleButton Creation Structure

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

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

Table 3.3.41-3 – ToggleButton Event Structures: A661_EVT_STATE_CHANGE

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

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

Table 3.3.41-4 – ToggleButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
ToggleState	uchar	8	A661_INNER_STATE_TOGGLE
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
LabelString	string	{32}+	A661_STRING
AlternateLabelString	string	{32}+	A661_STRING_ALTERNATE
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
AlternateFlag	**uchar**	**8**	**A661_ALTERNATE_FLAG**

3.3.42 TranslationContainer

Categories:

- Container

Description:
A TranslationContainer widget applies a translation transformation to a group of widgets. Widgets placed within a TranslationContainer have their coordinates referenced from TranslationX and TranslationY.

Restriction:
None

TranslationContainer Parameters are defined in Table 3.3.42-1.

Table 3.3.42-1 – TranslationContainer Parameters

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

TranslationContainer Creation Structure is defined in Table 3.3.42-2.

Table 3.3.42-2 – TranslationContainer Creation Structure

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

The TranslationContainer widget does not send any event.

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

Table 3.3.42-3 – TranslationContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
TranslationX, TranslationY	long x 2	32 x 2	A661_TRANSLATION_XY
TranslationX	long	32	A661_TRANSLATION_X
TranslationY	long	32	A661_TRANSLATION_Y

3.4 Widgets Added for Supplement 1

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

3.4.1 MapGrid

Categories:

- Map Management
- Graphical Representation

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

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

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

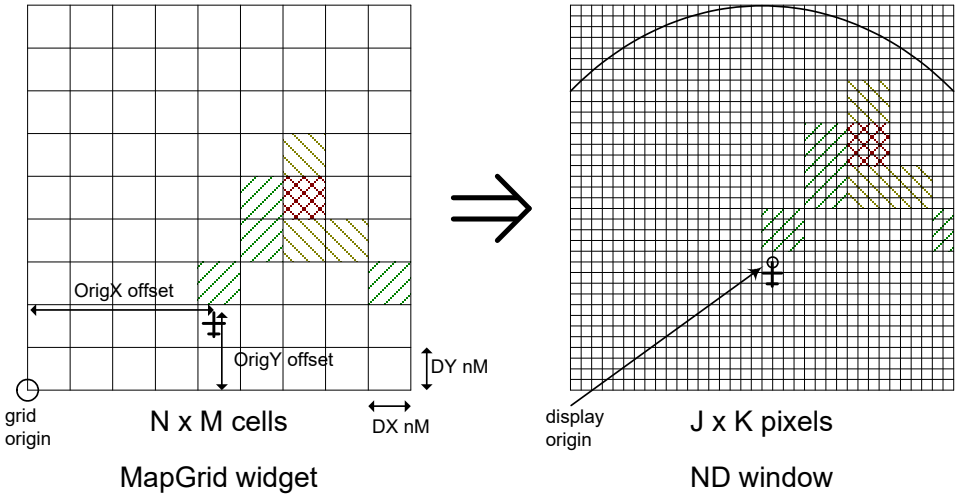


Figure 3.4.1 – Example MapGrid Rendering in ND window

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

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

MapGrid Parameters are defined in Table 3.4.1-1.

Table 3.4.1-1 – MapGrid Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAP_GRID
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Specific parameters		
CountX	D	Number of cells along the X axis in the array.
CountY	D	Number of cells along the Y axis in the array.
OffsetX	DR	Horizontal offset, in (fractional) cells, between the display reference point and the grid origin. If translation/rotation is done by the UA, this number is constant (typically one-half of CountX). (OffsetX, OffsetY) defines the point in the grid to be placed at the display origin (typically, the aircraft current location). The point may or may not be at a cell boundary, depending on the ratio of cell size to pixel size, chosen aircraft mock-up location, and whether translation for aircraft motion is implemented in the CDS or the UA.
OffsetY	DR	Vertical offset, in (fractional) cells, between the display reference point and the grid origin.
IncrementX	DR	Size of each individual cell in the X axis, in the real-world units defined by the containing Map Source. Support for the various Map Source Data Formats depends on the implementation. MapHorz_Source: A661_MDF_LAT_LONG: See Table 3.3.24-2b. A661_MDF_DIST_DIST: See Table 3.3.24-2b. A661_MDF_BRG_DIST_ACHDG: See Table 3.3.24-2b. A661_MDF_DIST_DIST_FIX: See Table 3.3.24-2b. A661_MDF_LEGACY See Table 3.3.24-2b. MapVert_Source: A661_MDF_X_DIST See Table 3.4.4-2b. A661_MDF_RELATIVE See Table 3.4.4-2b. A661_MDF_ABSOLUTE See Table 3.4.4-2b.

Parameters	Change	Description
IncrementY	DR	Size of each individual cell in the Y axis, in the real-world units defined by the containing Map Source. Support for the various Map Source Data Formats depends on the implementation. MapHorz_Source: A661_MDF_LAT_LONG: See Table 3.3.24-2b. A661_MDF_DIST_DIST: See Table 3.3.24-2b. A661_MDF_BRG_DIST_ACHDG: See Table 3.3.24-2b. A661_MDF_DIST_DIST_FIX: See Table 3.3.24-2b. A661_MDF_LEGACY See Table 3.3.24-2b. MapVert_Source: A661_MDF_Y_ALT See Table 3.4.4-2b. A661_MDF_RELATIVE See Table 3.4.4-2b. A661_MDF_ABSOLUTE See Table 3.4.4-2b.
BufferOfFillStyles	R	Buffer of Fill Style Indices. Buffer can be updated row-at-a-time or column-at-a-time.
MapSynchronization Number	R	See Section 3.2.8.4

MapGrid Creation Structure is defined in Table 3.4.1-2.

Table 3.4.1-2 – MapGrid Creation Structure

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

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

Table 3.4.1-3 – MapGrid Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
OffsetX, OffsetY	float x 2	32 x 2	A661_MAPGRID_OFFSET
IncrementX, IncrementY	float x 2	32 x 2	A661_MAPGRID_CELLSIZE
BufferOfFillStyles	N/A	{32}+	A661_BUFFER_OF_FILL_STYLES Refer to “MapGrid Parameter Structure Specifics” section below.
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4

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

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

3.4.1.1 MapGrid A661_ParameterStructure Specifics

Table 3.4.1.1-1 – A661_ParameterStructure_BufferOfFillStyles

Field	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
StartIndexX	ushort	16	Index (zero-based) of column to start storing data.
StartIndexY	ushort	16	Index (zero-based) of row to start storing data. Some CDS implementations may require that one of StartIndexX or StartIndexY be zero.
NumColumns	ushort	16	Number of columns being filled.
NumRows	ushort	16	Number of rows being filled. Some CDS implementations may require that one of NumColumns or NumRows be set to CountX or CountY, respectively. For some implementations the restriction may only be enforced if the other of NumColumns, NumRows is greater than one.
StepX	uchar	8	+1 or –1 (0xFF) May be set to 0 if NumColumns is 1.
StepY	uchar	8	+1 or –1 (0xFF) May be set to 0 if NumRows is 1. Some CDS implementations may suggest or require that StepX and/or StepY always be set to a specific value (such as +1).

Field	Type	Size (bits)	Description
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 **(see Section 3.1.3.3.1)**. 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 bits alignment.

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

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

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

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

3.4.1.2 Fill Style Index Values

See Section 3.1.3.3.1.

COMMENTARY

The actual fill styles used will depend on 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 may have its own palette. In others, all UAs may share a global fill palette.

3.4.2 ExternalSource

Categories:

None

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

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

  Note: the ExternalSource widget makes it necessary for the UA supplier and the integrator to define the specific method to bring video to the display.

Restriction:
None

ExternalSource Parameters are defined in Table 3.4.2-1.

Table 3.4.2-1 – ExternalSource Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EXTERNAL_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
MotionAllowed	**D**	**Capability to change PosX, PosY, SizeX, SizeY at runtime.**
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
SourceReference	DR	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	May control transparency, zoom/clip, etc.

ExternalSource Creation Structure is defined in Table 3.4.2-2.

Table 3.4.2-2 – ExternalSource Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EXTERNAL_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
MotionAllowed	**uchar**	8	**A661_FALSE** **A661_TRUE**
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SourceReference	ushort	16	
UnusedPad	N/A	16	0
SourceX	ulong	32	
SourceY	ulong	32	
SourceDX	ulong	32	
SourceDY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0

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
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
SourceX	ulong	32	A661_SOURCE_X
SourceY	ulong	32	A661_SOURCE_Y
SourceX, SourceY	ulong x 2	32 x 2	A661_SOURCE_XY
SourceDX	ulong	32	A661_SOURCE_DX
SourceDY	ulong	32	A661_SOURCE_DY
SourceDX, SourceDY	ulong x 2	32 x 2	A661_SOURCE_DXDY
SourceReference	**ushort**	**16**	**A661_SOURCE_REF**
StyleSet	ushort	16	A661_STYLE_SET

3.4.3 MapVert

Categories:

- Container
- Map Management

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

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

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
ParentIdent	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 interactive
PosX	DR	The X position of the widget reference point (screen coordinate system)
PosY	DR	The Y position of the widget reference point (screen coordinate system)
SizeX	DR	Area size X
SizeY	DR	Area size Y
RangeX	DR	Equivalent in Geographic coordinates, expressed in nm, of Area Size X
RangeY	DR	Equivalent in Geographic coordinates, expressed in feet, of Area Size Y
RefPosX	DR	Position X of the reference point, expressed in screen coordinates (hundredth of mm) from PosX
RefPosY	DR	Position Y of the reference point, expressed in screen coordinates (hundredth of mm) from PosY
RefGeoPosX	DR	Geo-referenced Position X of the reference point, expressed in nm
RefGeoPosY	DR	Geo-referenced Position Y of the reference point, expressed in feet
MapSynchronization Number	R	See Section 3.2.8.4

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

Table 3.4.3-2a – MapVert Creation Structure

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

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

Table 3.4.3-2b – MapVert Event Structures: A661_EVT_ITEM_SYNCHRONIZATION

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

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

Table 3.4.3-3 – MapVert Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
RangeX	fr(32768)	32	A661_RANGE_X
RangeY	long	32	A661_RANGE_Y
RangeX, RangeY	fr(32768) long	32 x 2	A661_RANGE_XY
RefPosX	long	32	A661_PRP_SCREEN_X
RefPosY	long	32	A661_PRP_SCREEN_Y
RefPosX, RefPosY	long x 2	32 x 2	A661_PRP_SCREEN_XY
RefGeoPosX	fr(32768)	32	A661_PRP_X
RefGeoPosY	long	32	A661_PRP_Y
RefGeoPosX, RefGeoPosY	fr(32768) long	32 x 2	A661_PRP_XY
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER See Section 3.2.8.4
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY

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 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 for the map UA to control the CDS sending this event. The X,Y position sent by the CDS is expressed in the MapVert_Source coordinate system.

Restriction:
The MapVert_Source should be under a MapVert widget **(directly or via a MaskContainer 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
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
MapDataFormatX	D	Relative: X position of Items expressed relative to Reference point. Absolute: X position of Items expressed in absolute value. X_Dist: X fixed position with respect to screen reference point (no motion compensation)
MapDataFormatY	D	Relative: Y position of Items expressed relative to Reference point. Absolute: Y position of Items expressed in absolute value. Y_Alt: Y fixed position with respect to screen reference point (no motion compensation)
EventFlag	DR	Indicates if the UA wants to receive the cursor position upon click, expressed in its coordinate system.

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

Table 3.4.4-2a – MapVert_Source Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPVERT_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
MapDataFormatX	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE A661_MDF_X_DIST
MapDataFormatY	uchar	8	A661_MDF_ABSOLUTE A661_MDF_RELATIVE A661_MDF_Y_ALT
EventFlag	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0

Based on the MapDataFormat setting, the X and Y parameters for MapVert_ItemList items can represent real world quantities such as Range and Altitude. Units are as given in the following table.

Note that when Relative Positioning is being used (for MIL items having the RelativePosition parameter), the X and Y parameters are expressed in screen coordinates and are encoded using data type “long” (32 bits signed), regardless of the MapDataFormat setting.

MapVert_MapDataFormat Structure is defined in Table 3.4.4-2b.

Table 3.4.4-2b – MapVert_MapDataFormat Values

Parameter	Value	Origin	Units of Measure	Type (Note 1)
MapDataFormatX	A661_MDF_ABSOLUTE	Absolute Reference: Position with respect to RefGeoPosX is: DataPosX-RefGeoPosX	nM	fr(32768)
MapDataFormatX	A661_MDF_RELATIVE	Relative Reference: RefGeoPosX The Position with respect to RefGeoPosX is: DataPosX	nM	fr(32768)
MapDataFormatX	A661_MDF_X_DIST	RefPosX	nM	float
MapDataFormatY	A661_MDF_ABSOLUTE	Absolute Reference: Position with respect to RefGeoPosY is: DataPosY-RefGeoPosY	feet	long
MapDataFormatY	A661_MDF_RELATIVE	Relative Reference: RefGeoPosY The Position with respect to RefGeoPosY is: DataPosY	feet	long
MapDataFormatY	A661_MDF_Y_ALT	RefPosY	feet	float

Note:

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

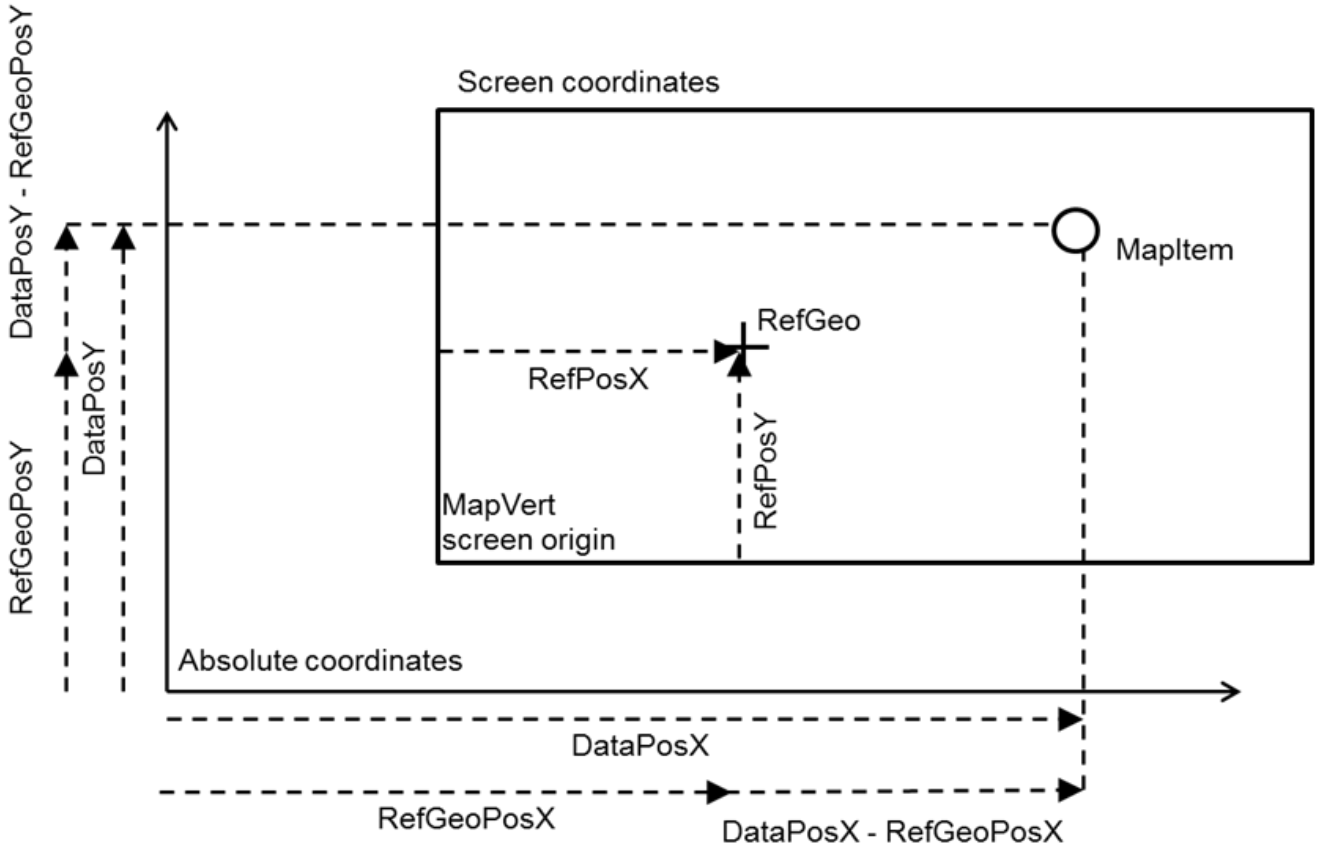


Figure 3.4.4-1 – Illustration of MapDataFormatX/Y = A661_MDF_ABSOLUTE

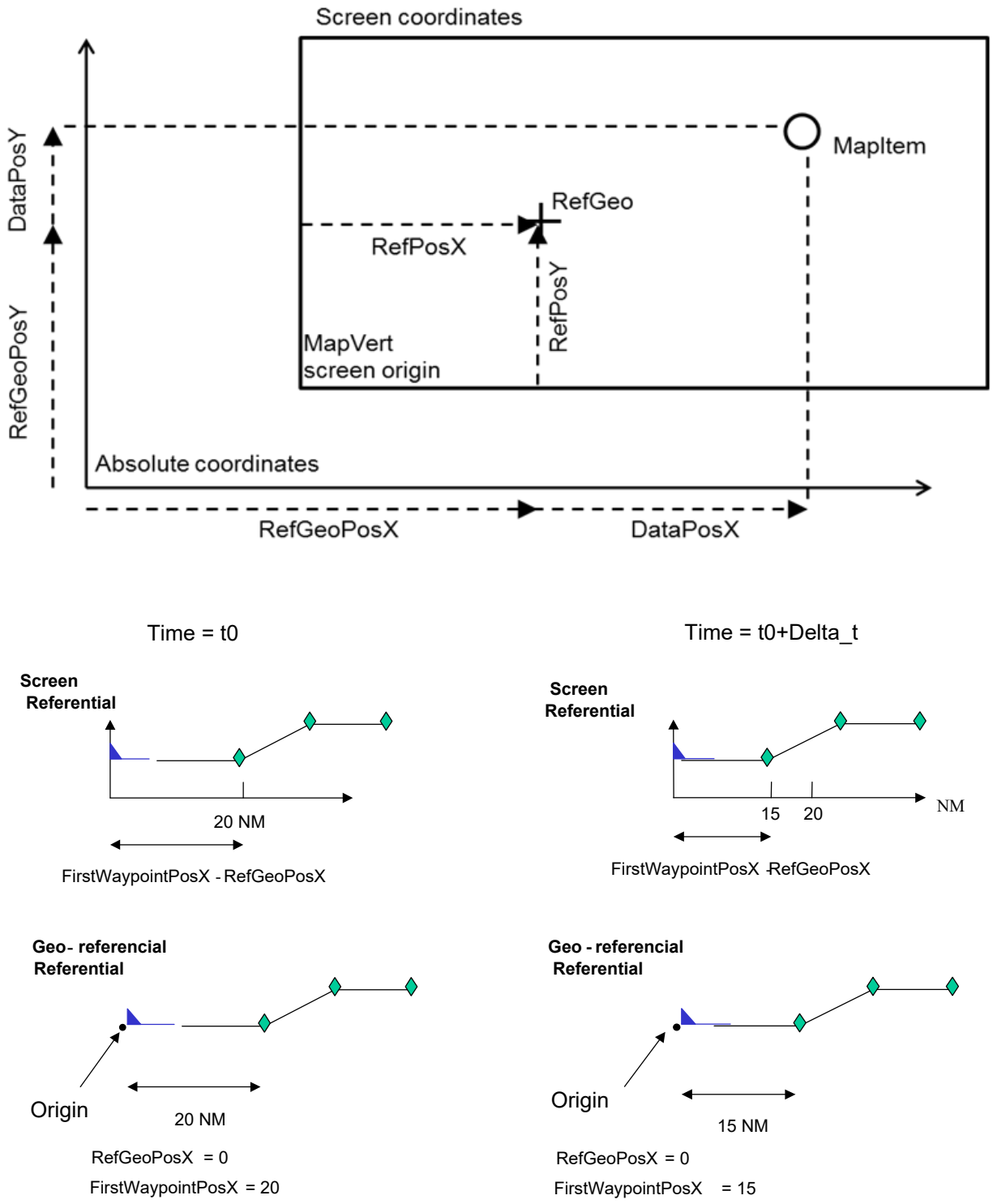


Figure 3.4.4-2 – Illustration of MapDataFormatX/Y = A661_MDF_RELATIVE

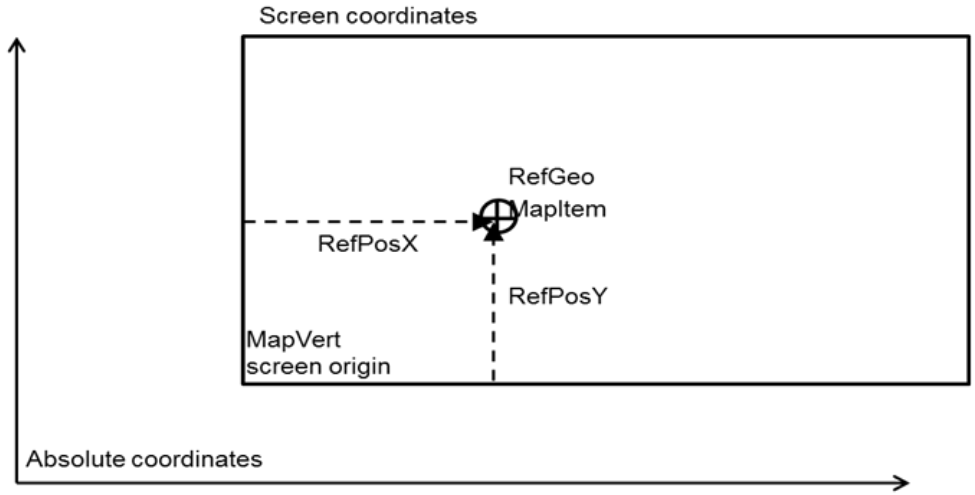


Figure 3.4.4-3 – Illustration of MapDataFormatX/Y = A661_MDF_X_DIST/A661_MDF_Y_ALT

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	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

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

Table 3.4.4-4 – MapVert_Source Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
EventFlag	uchar	8	A661_EVENT_FLAG

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 or BlockedBufferOfItems as Parameter_Ident. An item should be modified in its entirety. For instance, the X coordinate of a symbol item cannot be changed by itself.

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

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

Restriction:
A MapVert_ItemList must be in a MapVert_Source container.

MapVert_ItemList Parameters is defined in Table 3.4.5-1.

Table 3.4.5-1 – MapVert_ItemList Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPVERT_ITEMLIST
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
Specific parameters		
MaxNumberOfItem	D	Maximum number of items that the UA can address under the MapVert_ItemList.
MaxItemSize	**D**	**Hint provided by the UA that indicates to the CDS what is the expected maximum size in 32-bit words of an item this MapVert_ItemList may contain.** **If set to 0, no hint is provided.** **Refer to Section 3.4.5.2.1.**
BufferOfItems	R	Buffer of the Map Items
MapSynchronizationNumber	R	See Section 3.2.8.4
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
BlockedBufferOfItems	R	Buffer of map items that can span multiple messages in order to make an atomic update to a map item list that cannot fit in a single message.

MapVert_ItemList Creation Structure is defined in Table 3.4.5-2a.

Table 3.4.5-2a – MapVert_ItemList Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPVERT_ITEMLIST
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfItem	ushort	16	
MaxItemSize	**uchar**	**8**	
UnusedPad	**N/A**	**8**	**0**

MapVert_ItemList Event Structures: A661_EVT_SELECTION is defined in Table 3.4.5-2b.

Table 3.4.5-2b – MapVert_ItemList Event Structures: A661_EVT_SELECTION

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

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

Note: The structure of this event is different than other A661_EVT_SELECTION events in that it does not contain a pad.

Table 3.4.5-3 – MapVert_ItemList Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
BufferOfItems	N/A	{32}	A661_BUFFER_OF_MAPVERT_ITEMS Refer to “MapVert_ItemList A661_ParameterStructure Specifics” Section 3.4.5.2, especially Section 3.4.5.2.2.
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER See Section 3.2.8.4
EntryValidation	uchar	8	A661_ENTRY_VALID
BlockedBufferOfItems	N/A	{32}	A661_BLOCKED_BUFFER_OF_MAPVERT_ITEMS Refer to “MapVert_ItemList A661_ParameterStructure Specifics” Section 3.4.5.2, especially Section 3.4.5.2.3.

3.4.5.1 MapVert_ItemList Standard Items Description

This section describes all the Item structures.

Table 3.4.5.1 – MapVert_ItemList Standard Items Description

Name of Item	Size in Words	Function
FILLED_POLY_START	3	This Item is used to signify the start of a closed, filled polygon definition. It holds X/Y parameters (similar to LINE_START) but also has a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items define the vertices and edges of a polygon that is closed and filled with the indicated fill style. The end of the polygon definition is indicated when the EndFlag in the last LINE_SEGMENT item in the group is set to True.
FILLED_POLY_START_INTERACTIVE	**3**	**This Item is used to signify the start of a closed, filled interactive polygon definition. It holds X/Y parameters (similar to LINE_START) but also has a Fill Style Index. The X/Y parameters of this Item and the following LINE_SEGMENT Items define the vertices and edges of a polygon that is closed and filled with the indicated fill style. The end of the polygon definition is indicated when the EndFlag in the last LINE_SEGMENT item in the group is set to True.**
ITEM_STYLE	2	For drawing any symbol, line or text legend the CDS must apply the last defined ITEM_STYLE found in the Map Item List. If no ITEM_STYLE has been provided, the CDS will apply the default ITEM_STYLE. The list of available ITEM_STYLE enumerations and their associated graphical representations are implementation dependent.
ITEM_SYNCHRONIZATION	3	This item has been defined to attach frame data to symbology expressing the context of the computation or the rendering for the symbology frame. This data is sent in ARINC 661 in order to avoid synchronization issues between the symbology frame and the attached information (e.g., mode, range, MRP, etc.). This item can be used to pass synchronization information from the application owning a MapVert Item List to the application owning the ancestor MapVert.
LEGEND	**5**	**This Item is used to represent Legend Strings. Section 3.2.9.5 describes how the LegendString is managed.**
LEGEND_ANCHOR	3	This Item is used to specify the position of a LEGEND not attached to a symbol.

Name of Item	Size in Words	Function
LEGEND_ANCHOR_ROTATED	5	Same as LEGEND_ANCHOR but includes Legend orientation and control parameters.
LEGEND_COMBO	5	This item allows a string to be defined as any combination of LEGEND, LEGEND_POP_UP, and LEGEND_HIGHLIGHT in a single map item. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_READOUT	2	This Item is used to represent a formatted numeric value. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_READOUT_FORMAT_STRING	5	For formatting any LEGEND_READOUT or LEGEND_READOUT_HIGHLIGHT the CDS must apply the last defined LEGEND_READOUT_FORMAT_STRING found in the Map Item List. If no LEGEND_READOUT_FORMAT_STRING has been provided, the CDS will apply a default LEGEND_READOUT_FORMAT_STRING. Section 3.2.9.5 describes how the LegendString is managed.
LEGEND_READOUT_COMBO	3	This item allows a string to be defined as any combination of LEGEND_READOUT, popped-up LEGEND_READOUT, and highlighted LEGEND_READOUT in a single map item. Section 3.2.9.5 describes how the LegendString is managed.
LINE_START	3	This Item is used to signify the start of a line. It has only X/Y parameters, interpreted by the CDS depending on the MapVert_Source DataFormat. The LINE_START, when followed by one or more LINE_SEGMENTitems, defines a line sequence to be rendered. See Appendix E for an example of a Line Item group.
LINE_START_INTERACTIVE	3	This Item is used to signify the start of an interactive line. It has only X/Y parameters, interpreted by the CDS depending on the MapVert_Source DataFormat. The LINE_START, when followed by one or more LINE_SEGMENTitems, defines a line sequence to be rendered. See Appendix E for an example of a Line Item group.
LINE_SEGMENT	3	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 has an EndFlag, set to True if it is the last item of a Line Item (or polygon) group.

Name of Item	Size in Words	Function
LINE_SEGMENT_INTERACTIVE	**3**	**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 has an EndFlag, set to True if it is the last item of a Line Item (or polygon) group.**
NOT_USED	**1**	This Item is used when the Item is to be discarded by the CDS. There is no effect on interpretation of subsequent Items.
SYMBOL_GENERIC	**4**	This Item represents a basic symbol having X/Y position, symbol type and EndFlag parameters. The EndFlag is set to True if no Legend is attached to the Symbol (in effect, this indicates the end of the SYMBOL Item). If the EndFlag is False, a Legend is to be drawn with the Symbol, and one or more LEGEND items will follow the SYMBOL item. SYMBOL items (and associated LEGENDs, if any) can be embedded in a Line Item group, but not in a Polygon group. See Appendix E for a description of this.
SYMBOL_GENERIC_INTERACTIVE	**4**	**This Item represents a basic interactive symbol having X/Y position, symbol type and EndFlag parameters. The EndFlag is set to True if no Legend is attached to the Symbol (in effect, this indicates the end of the SYMBOL Item). If the EndFlag is False, a Legend is to be drawn with the Symbol, and one or more LEGEND items will follow the SYMBOL item. SYMBOL items (and associated LEGENDs, if any) can be embedded in a Line Item group, but not in a Polygon group. See Appendix E for a description of this.**
SYMBOL_ROTATED	**5**	Same as SYMBOL_GENERIC (i.e., Function descriptions for SYMBOL_GENERIC apply to SYMBOL_ROTATED as well), except an orientation parameter is added.
SYMBOL_ROTATED_INTERACTIVE	**5**	**Same as SYMBOL_GENERIC_INTERACTIVE (i.e., Function descriptions for SYMBOL_GENERIC_INTERACTIVE apply to SYMBOL_ROTATED as well), except an orientation parameter is added.**
SYMBOL_RUNWAY	**4**	Specific Symbol. It represents an airport runway, with a runway length parameter. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.

Name of Item	Size in Words	Function
SYMBOL_RUNWAY_INTERACTIVE	4	Specific interactive Symbol. It represents an airport runway, with a runway length parameter. This Item has an EndFlag, set to True if no LEGEND is attached to the Symbol.
SYMBOL_TARGET	5	Same as SYMBOL_GENERIC (i.e., Function descriptions for SYMBOL_GENERIC apply to SYMBOL_TARGET as well), except part of the symbol can be rotated and part of the symbol has variable length illustrating Velocity/Distance/Altitude.
SYMBOL_TARGET_INTERACTIVE	5	Same as SYMBOL_GENERIC_INTERACTIVE (i.e., Function descriptions for SYMBOL_GENERIC_INTERACTIVE apply to SYMBOL_TARGET_INTERACTIVE as well), except part of the symbol can be rotated and part of the symbol has variable length illustrating Velocity/Distance/Altitude.
SYMBOL_RECTANGLE	5	Specific symbol. It represents a rectangle, filled with the indicated fill style, which may be “no fill.”
SYMBOL_RECTANGLE_INTERACTIVE	5	Specific interactive symbol. It represents a rectangle, filled with the indicated fill style, which may be “no fill.”
TRIANGLE_STRIP_START	5	This Item is used to signify the start of a closed, filled polygon defined by a series of triangle strips.
TRIANGLE_STRIP_START_INTERACTIVE	5	This Item is used to signify the start of a closed, filled interactive A661_SYMBOL_PARKABLE polygon defined by a series of triangle strips.
TRIANGLE_FAN_START	5	This Item is used to signify the start of a closed, filled polygon defined by a series of triangles arranged in a fan.
TRIANGLE_FAN_START_INTERACTIVE	5	This Item is used to signify the start of a closed, filled interactive polygon defined by a series of triangles arranged in a fan.
TRIANGLE_SEGMENT	3	Defines a single vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_INTERACTIVE	3	Defines a single vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_DOUBLE	5	Defines two vertices of a Triangle Strip or Triangle Fan.
TRIANGLE_SEGMENT_DOUBLE_ INTERACTIVE	5	Defines two vertices of a Triangle Strip or Triangle Fan.
TRIANGLE_END	3	Defines the last vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_END_INTERACTIVE	3	Defines the last vertex of a Triangle Strip or Triangle Fan.
TRIANGLE_END_DOUBLE	5	Defines the last two vertices of a Triangle Strip or Triangle Fan.

Name of Item	Size in Words	Function
TRIANGLE_END_DOUBLE_INTERACTIVE	5	Defines the last two vertices of a Triangle Strip or Triangle Fan.
DRAW_LINE_TO_CURSOR	3	This item specifies a point from which a line will be drawn to the current cursor position.

Error Handling For MIL Item EndFlag

The preceding table describes how the EndFlag should be used by the UA when constructing MapItem Lists. Associated error detection and handling (e.g., a missing EndFlag) is implementation dependent.

Positioning Of Map Symbology

For SYMBOL_xxx map items, the X/Y coordinate parameters provided by the UA constitute a symbol positioning reference point. Generally, this point is expressed in real world coordinates. The OEM and CDS developer must agree on how the CDS should position the corresponding graphics symbols on the map display. For some map items it will be appropriate to display the symbol so it is centered on the real world position represented by the X/Y reference point. For other items, the CDS will place symbols based on the functional meaning of the map item (e.g., when using SYMBOL_RUNWAY the UA will usually set the X/Y reference point to the airport runway threshold position). The SymbolType parameter and ITEM_STYLE are also used to control positioning of symbols on the map display.

Relative Positioning Of Map Symbology

Placement of map items is generally determined using a real world coordinate system. In order to support decoration of symbols, Relative Positioning is also available for several item types. This allows graphics items to be placed on a map relative to a Symbol or some other fixed location, using screen coordinates instead of real world coordinates. The positioning of these relative elements does not change based on the parameters used to calculate coordinate transformations. Map Item List Items that have the RelativePosition parameter can be positioned via Relative Positioning.

For an example of SYMBOL and LEGEND placement using relative screen coordinates, see Figure 3.3.22.1.

Overlapping Active Areas

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

3.4.5.2 MapVert_ItemList A661_ParameterStructure Specifics

This section describes the A661_ParameterStructure_BufferOfItems for MapVert_ItemList.

3.4.5.2.1 Item Structures

This section describes the MapVert_ItemList 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.

COMMENTARY

CDS implementers will want to bound the amount of memory used by the Map Item List. This could be determined by either observing that there is a maximum size of a MIL item **or by using the MaxItemSize parameter**.

Maximum size for any MIL can be deduced from Table 3.4.5.1.
The memory required by the CDS to store the MIL could be the value of the MaxNumberOfItem parameter in the MapVert_ItemList creation structure multiplied by the current maximum MIL item size.

The UA can also propose a MaxItemSize parameter. This parameter can be used to specify the actual maximum item size for a MIL or any optimized size for the map item. (average size, average size with margin, etc.). The memory required by the CDS to store the MIL could be the value of the MaxNumberOfItem parameter in the MapVert_ItemList creation structure multiplied by the MaxItemSize parameter. In case a larger item is received, CDS behavior is implementation dependent. In case the parameter is set to 0, it is up to the CDS to manage memory allocation.

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)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_STYLE
UnusedPad	N/A	8	0
ItemStyleSet	ushort	16	
UnusedPad	N/A	16	0

3.4.5.2.1.2 Legend_Anchor

Legend_Anchor is defined in Table 3.4.5.2.1.2.

Table 3.4.5.2.1.2 – Legend_Anchor

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

3.4.5.2.1.3 Legend, Legend_Highlight, and Legend_Pop_Up

Legend, Legend_Highlight, and Legend_Pop_Up are 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)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND A661_LEGEND_HIGHLIGHT 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 on 32 bits. Table 3.4.4-2b defines the proper string termination.

3.4.5.2.1.4 Line_Start and Line_Start_Interactive

Line_Start and Line_Start_Interactive is defined in Table 3.4.5.2.1.4.

Table 3.4.5.2.1.4 – Line_Start and Line_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_START A661_LINE_START_INTERACTIVE
UnusedPad	N/A	8	0
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

3.4.5.2.1.5 Line_Segment and Line_Segment_Interactive

Line_Segment and Line_Segment_Interactive is defined in Table 3.4.5.2.1.5.

Table 3.4.5.2.1.5 – Line_Segment and Line_Segment_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LINE_SEGMENT A661_LINE_SEGMENT_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

3.4.5.2.1.6 Not_Used

Not_Used is defined in Table 3.4.5.2.1.6.

Table 3.4.5.2.1.6 – Not_Used

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

3.4.5.2.1.7 Symbol_Generic and Symbol_Generic_Interactive

Symbol_Generic and Symbol_Generic_Interactive is defined in Table 3.4.5.2.1.7.

Table 3.4.5.2.1.7 – Symbol_Generic and Symbol_Generic_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_GENERIC A661_SYMBOL_GENERIC_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	SYMBOL_WAYPOINT SYMBOL_AIRPORT SYMBOL_VOR SYMBOL_VORDME
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	uchar	8	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.4.4-2b

3.4.5.2.1.8 Symbol_Runway and Symbol_Runway_Interactive

Symbol_Runway and Symbol_Runway_Interactive is defined in Table 3.4.5.2.1.8.

Table 3.4.5.2.1.8 – Symbol_Runway and Symbol_Runway_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RUNWAY A661_SYMBOL_RUNWAY_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.4.4-2b
Length	fr(32768)	32	Length of runway (in feet)

3.4.5.2.1.9 Filled_Poly_Start and Filled_Poly_Start_Interactive

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 notification. The OEM free data field may include the ItemIndex, etc., to identify the error further.

Filled_Poly_Start and Filled_Poly_Start_Interactive is defined in Table 3.4.5.2.1.9.

Table 3.4.5.2.1.9 – Filled_Poly_Start and Filled_Poly_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_FILLED_POLY_START A661_FILLED_POLY_START_INTERACTIVE
FillStyleIndex	uchar	8	See Section 3.1.3.3.1
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

3.4.5.2.1.10 Item_Synchronization

Item_Synchronization is defined in Table 3.4.5.2.1.10.

Table 3.4.5.2.1.10 – Item_Synchronization

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_ITEM_SYNCHRONIZATION
DataType	uchar	8	For example: VD_MODE_RANGE MRP latitude/longitude…
SynchronizationData 1st word		32	Data is implementation dependent.
SynchronizationData 2nd word		32	Data is implementation dependent.

3.4.5.2.1.11 Symbol_Rotated and Symbol_Rotated_Interactive

Symbol_Rotated and Symbol_Rotated_Interactive is defined in Table 3.4.5.2.1.11.

Table 3.4.5.2.1.11 – Symbol_Rotated and Symbol_Rotated_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_ROTATED A661_SYMBOL_ROTATED_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	For example, SYMBOL_AIRCRAFT
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
UnusedPad	N/A	8	0
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.4.4-2b
Orientation	fr(180)	32	Orientation of Symbol (counter-clockwise is positive orientation) relative to upright axis of the MapVert display (e.g., positive altitude direction). Orientation is not adjusted by the CDS to account for the scaling of the RangeX and RangeY in the MapVert widget.

3.4.5.2.1.12 Triangle_Strip_Start and Triangle_Strip_Start_Interactive

The Triangle_Strip_Start, Triangle_Segment, Triangle_Segment_Double, Triangle_End, and Triangle_End_Double MapVert_ItemList Items are meant to define triangle strips, similar to those defined in SDL Symbols.

Each Triangle Strip is made up of one Triangle_Strip_Start, any number of Triangle_Segment and Triangle_Segment_Double items, and one Triangle_End or Triangle_End_Double. The Triangle_Segment_Double and Triangle_End_Double items exist to minimize MapVert_ItemList size and should be used whenever possible. See the illustration below for more details.

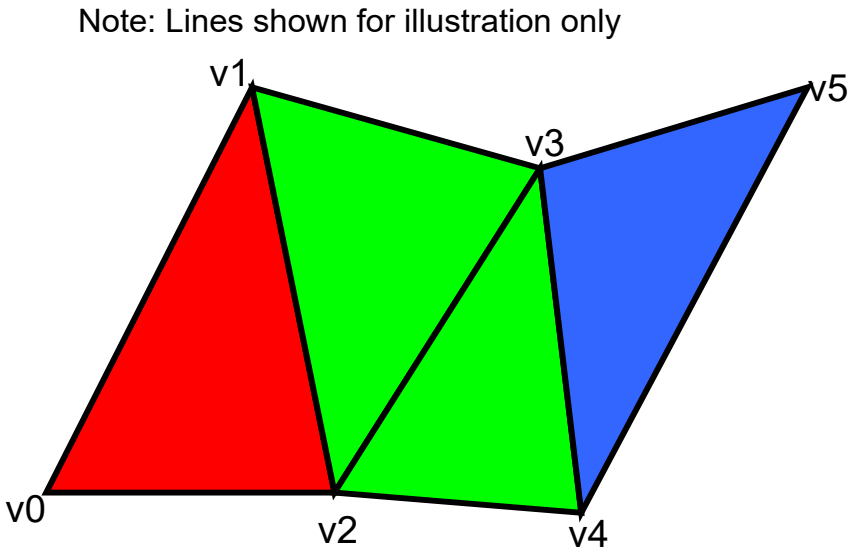


Figure 3.4.5.2.1.12 – Triangle_Strip

Triangle_Strip_Start and Triangle_Strip_Start_Interactive is defined in Table 3.4.5.2.1.12.

Table 3.4.5.2.1.12 – Triangle_Strip_Start and Triangle_Strip_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_STRIP_START A661_TRIANGLE_STRIP_START_INTERACTIVE
UnusedPad	N/A	8	0
X1	Dependent on MapDataFormat	32	First Coordinate of first vertex. See Table 3.4.4-2b
Y1	Dependent on MapDataFormat	32	Second Coordinate of first vertex. See Table 3.4.4-2b
X2	Dependent on MapDataFormat	32	First Coordinate of second vertex. See Table 3.4.4-2b
Y2	Dependent on MapDataFormat	32	Second Coordinate of second vertex. See Table 3.4.4-2b

3.4.5.2.1.13 Triangle_Segment and Triangle_Segment_Interactive

The fill of the triangle completed by this item is defined by the FillStyleIndex parameter.

Triangle_Segment and Triangle_Segment_Interactive is defined in Table 3.4.5.2.1.13.

Table 3.4.5.2.1.13 – Triangle_Segment and Triangle_Segment_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_SEGMENT A661_TRIANGLE_SEGMENT_ INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangle completed by this point see Section 3.1.3.3.1
X	Dependent on MapDataFormat	32	First coordinate of vertex. See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	Second coordinate of vertex. See Table 3.4.4-2b

3.4.5.2.1.14 Triangle_Segment_Double and Triangle_Segment_Double_Interactive

The fill of the triangles completed by this item is defined by the FillStyleIndex parameter.

Triangle_Segment_Double and Triangle_Segment_Double_Interactive is defined in Table 3.4.5.2.1.14.

Table 3.4.5.2.1.14 – Triangle_Segment_Double and Triangle_Segment_Double_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_SEGMENT_DOUBLE A661_TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangles completed by these points see Section 3.1.3.3.1
X1	Dependent on MapDataFormat	32	First Coordinate of first vertex. See Table 3.4.4-2b
Y1	Dependent on MapDataFormat	32	Second Coordinate of first vertex. See Table 3.4.4-2b
X2	Dependent on MapDataFormat	32	First Coordinate of second vertex. See Table 3.4.4-2b
Y2	Dependent on MapDataFormat	32	Second Coordinate of second vertex. See Table 3.4.4-2b

3.4.5.2.1.15 Triangle_End and Triangle_End_Interactive

The fill of the triangle completed by this item is defined by the FillStyleIndex parameter.

Triangle_End and Triangle_End_Interactive is defined in Table 3.4.5.2.1.15.

Table 3.4.5.2.1.15 – Triangle_End and Triangle_End_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_END A661_TRIANGLE_END_INTERACTIVE
FillStyleIndex	uchar	8	Fill for the triangle completed by this point see Section 3.1.3.3.1.

Name	Type	Size (bits)	Description and Value/Range
X	Dependent on MapDataFormat	32	First coordinate of vertex. See Table 3.4.4-2b.
Y	Dependent on MapDataFormat	32	Second coordinate of vertex. See Table 3.4.4-2b.

3.4.5.2.1.16 Triangle_End_Double and Triangle_End_Double_Interactive

The fill of the triangles completed by this item is defined by the FillStyleIndex parameter.

Triangle_End_Double and Triangle_End_Double_Interactive is defined in Table 3.4.5.2.1.16.

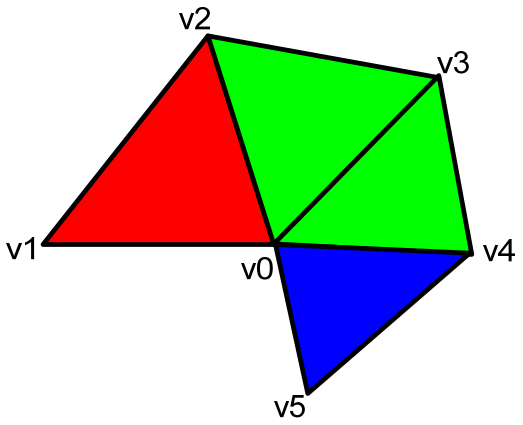
Table 3.4.5.2.1.16 – Triangle_End_Double and Triangle_End_Double_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_END_DOUBLE **A661_TRIANGLE_END_DOUBLE_INTERACTIVE**
FillStyleIndex	uchar	8	**Fill for the triangles completed by these points see Section 3.1.3.3.1.**
X1	Dependent on MapDataFormat	32	First Coordinate of first vertex. See Table 3.4.4-2b.
Y1	Dependent on MapDataFormat	32	Second Coordinate of first vertex. See Table 3.4.4-2b.
X2	Dependent on MapDataFormat	32	First Coordinate of second vertex. See Table 3.4.4-2b.
Y2	Dependent on MapDataFormat	32	Second Coordinate of second vertex. See Table 3.4.4-2b.

3.4.5.2.1.17 Triangle_Fan_Start and Triangle_Fan_Start_Interactive

The **Triangle_Fan_Start** (along with **Triangle_Segment**, **Triangle_Segment_Double**, **Triangle_End**, and **Triangle_End_Double**) MapVert_ItemList item is meant to define triangle fans, similar to those defined in SDL Symbols. Each Triangle Fan is made up of one **Triangle_Fan_Start**, any number of **Triangle_Segment** and **Triangle_Segment_Double** items, and one **Triangle_End** or **Triangle_End_Double**. The **Triangle_Segment_Double** and **Triangle_End_Double** items exist to minimize MapVert_ItemList size and should be used whenever possible.

Note: Black lines shown for illustration only



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

Figure 3.4.5.2.1.17 – Triangle_Fan_Start

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

Triangle_Fan_Start and Triangle_Fan_Start_Interactive is defined in Table 3.4.5.2.1.17.

Table 3.4.5.2.1.17 – Triangle_Fan_Start and Triangle_Fan_Start_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_TRIANGLE_FAN_START A661_TRIANGLE_FAN_START_INTERACTIVE
UnusedPad	N/A	8	0
X1	Dependent on MapDataFormat	32	First Coordinate of first vertex. See Table 3.4.4-2b.
Y1	Dependent on MapDataFormat	32	Second Coordinate of first vertex. See Table 3.4.4-2b.
X2	Dependent on MapDataFormat	32	First Coordinate of second vertex. See Table 3.4.4-2b.
Y2	Dependent on MapDataFormat	32	Second Coordinate of second vertex. See Table 3.4.4-2b.

Triangle_Segment, **Triangle_Segment_Double**, **Triangle_End**, and **Triangle_End_Double** are already defined (this was done in conjunction with **Triangle_Strip_Start**).

The CDS will process subsequent **Triangle_Segment**, **Triangle_Segment_Double**, **Triangle_End**, and **Triangle_End_Double** items appropriately based on the type of triangle start item that preceded them. That is, if a **Triangle_Fan_Start** item begins a sequence of triangle items, the first and third vertex of the previous triangle is used as the base for the triangle that follows. If a **Triangle_Strip_Start** item begins a sequence of triangle items, the second and third vertex of the previous triangle is used as the base for the triangle that follows.

3.4.5.2.1.18 Legend_Anchor_Rotated

Legend_Anchor_Rotated allows the UA to draw a legend on a map at a fixed orientation with respect to the upright axis of the MapVert display.

Note: The alignment of the text (its position relative to the Legend_Anchor_Rotated X/Y point) can be defined by the UA through the ITEM_STYLE Item.

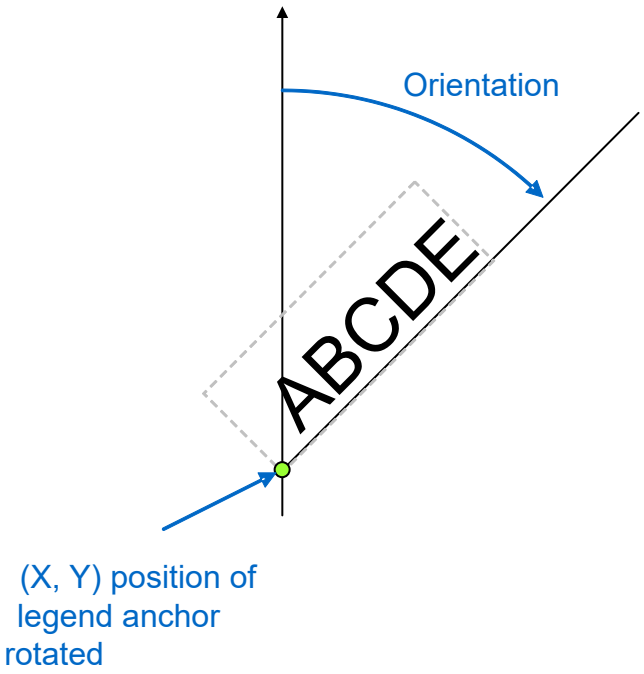


Figure 3.4.5.2.1.18 – Legend_Anchor_Rotated

Legend_Anchor_Rotated is defined in Table 3.4.5.2.1.18.

Table 3.4.5.2.1.18 – Legend_Anchor_Rotated

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_ANCHOR_ROTATED
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Orientation	fr(180)	32	Orientation of Legend (clockwise is positive rotation) relative to upright axis of the MapVert display (e.g., positive altitude direction).

Name	Type	Size (bits)	Description and Value/Range
LegendFlip	uchar	8	A661_FALSE A661_TRUE Defines whether the legend is flipped over when the map rotation makes the text inverted.
UnusedPad	N/A	24	0

3.4.5.2.1.19 Draw_Line_To_Cursor

Draw_Line_To_Cursor allows the UA to specify a point from which a line will be drawn to the current location of the cursor. The appearance of the line can be defined by the UA through the ITEM_STYLE Item.

The CDS behavior when there are multiple cursors inside the associated MapHorz display area is implementation dependent.

Draw_Line_To_Cursor is defined in Table 3.4.5.2.1.19.

Table 3.4.5.2.1.19 – Draw_Line_To_Cursor

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_DRAW_LINE_TO_CURSOR
RelativePosition	uchar	8	A661_FALSE A661_TRUE When RelativePosition is true then X and Y correspond to a position in screen units relative to the last symbol defined in the MapSource coordinate system.
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b

3.4.5.2.1.20 Symbol_Target and Symbol_Target_Interactive

Symbol_Target is defined in Table 3.4.5.2.1.20.

Table 3.4.5.2.1.20 – Symbol_Target and Symbol_Target_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_TARGET A661_SYMBOL_TARGET_INTERACTIVE
EndFlag	uchar	8	A661_TRUE A661_FALSE
SymbolType	ushort	16	EXAMPLES: AIR_FRIEND_TRACK AIR_SUSPECT_TRACK
Length	ushort	16	Velocity/Distance / Altitude illustrated by variable length part of the symbol (Knots/Nautical Miles/Feet). The choice of units is an OEM decision. The units for the MIL instance can be determined by the value of the SymbolType parameter, or in some other way.

Name	Type	Size (bits)	Description and Value/Range
X	Dependent on MapDataFormat	32	First coordinate of symbol See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol See Table 3.3.24-2b
Orientation	fr(180)	32	Orientation of Symbol (counter-clockwise is positive orientation) relative to upright axis of the MapVert display (e.g., positive altitude direction). Orientation is not adjusted by the CDS to account for the scaling of the RangeX and RangeY in the MapVert widget.

3.4.5.2.1.21 Symbol_Rectangle and Symbol_Rectangle_Interactive

Symbol_Rectangle and Symbol_Rectangle_Interactive is defined in Table 3.4.5.2.1.21.

Table 3.4.5.2.1.21 – Symbol_Rectangle and Symbol_Rectangle_Interactive

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_SYMBOL_RECTANGLE A661_SYMBOL_RECTANGLE_INTERACTIVE
FillStyleIndex	uchar	8	See Section 3.1.3.3.1.
X	Dependent on MapDataFormat	32	First coordinate of symbol. See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	Second coordinate of symbol. See Table 3.3.24-2b.
MajorSideLength	fr(32768)	32	Length of the Major Side in nautical miles
AxisRatio	fr(1)	16	Minor Side length divided by Major Side Length. Must be greater than zero.
Orientation	fr(180)	16	Orientation of Major Side (counter-clockwise is positive orientation) relative to upright axis of the MapVert display (e.g., positive altitude direction). Orientation is not adjusted by the CDS to account for the scaling of the RangeX and RangeY in the MapVert widget.

3.4.5.2.1.22 Legend_Combo

Legend_Combo allows a string to be defined as any combination of LEGEND, LEGEND_POP_UP, and LEGEND_HIGHLIGHT in a single map item. This allows fewer maps items to be used to define the same behavior.

Table 3.4.5.2.1.22-1 – Legend_Combo

LegendType	Equivalent to
A661_LEGEND_ONLY	One A661_LEGEND item
A661_POPUP_ONLY	One A661_LEGEND_POP_UP item
A661_HIGHLIGHT_ONLY	One A661_LEGEND_HIGHLIGHT item
A661_LEGEND_AND_POPUP	One A661_LEGEND item One A661_LEGEND_POP_UP item With the same String
A661_LEGEND_AND_HIGHLIGHT	One A661_LEGEND item One A661_LEGEND_HIGHLIGHT item With the same String

LegendType	Equivalent to
A661_POPUP_AND_HIGHLIGHT	One A661_LEGEND_POP_UP item One A661_LEGEND_HIGHLIGHT item With the same String
A661_LEGEND_AND_POPUP_AND_HIGHLIGHT	One A661_LEGEND item One A661_LEGEND_POP_UP item One A661_LEGEND_HIGHLIGHT item With the same String

Legend_Combo is defined in Table 3.4.5.2.1.22-2

Table 3.4.5.2.1.22-2 – Legend_Combo

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_COMBO
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendTypes	uchar	8	A661_LEGEND_ONLY A661_POPUP_ONLY A661_HIGHLIGHT_ONLY A661_LEGEND_AND_POPUP A661_LEGEND_AND_HIGHLIGHT A661_POPUP_AND_HIGHLIGHT A661_LEGEND_AND_POPUP_AND_HIGHLIGHT Value of 0 is reserved
LegendString	{uchar}+ (not 'string' because it might not be null terminated)	{8}+	Maximum of 15 characters. Section 3.2.9.5 defines the proper string termination and padding.

3.4.5.2.1.23 Legend_Readout_Format_String

Similar to the ItemStyle MapItem, for formatting the Legend_Readout the CDS must apply the last defined Legend_Readout_Format_String found in the Map Item List. If no A661_LEGEND_READOUT_FORMAT_STRING has been provided, the CDS will apply a default Legend_Readout_Format_String. The way the ReadoutFormat of this MapItem work is specified in Section 3.2.6 "Formatted Numeric Values.".

Note that this MapItem does not prevent the use of an ItemStyle on the same Legend_Readouts. For example, a Legend_Readout can have both a ReadoutFormat (to format its numeric value) and an ItemStyle (to apply a style to the corresponding Legend_Readouts).

Table 3.4.5.2.1.23 – Legend_Readout_Format_String

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT_FORMAT_STRING
EndFlag	uchar	8	A661_TRUE A661_FALSE
ReadoutFormat	{uchar}+ (not 'string' because it might not be null terminated)	{32}+	Max 16 characters. Section 3.2.9.5 defines the proper string termination and padding.

3.4.5.2.1.24 Legend_Readout

The MapItem Legend_Readout simplify the presentation of numbers (Number or index of Waypoints, Runway heading, etc.) use cases in the point of view of the UA.

Table 3.4.5.2.1.24 – Legend_Readout

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT
EndFlag	uchar	8	A661_TRUE A661_FALSE
ReadoutValue	float	32	

The formatting of this MapItem applies the ReadoutFormat defined in the last Legend_Readout_Format_String MapItem.

This MapItem can be mixed with Legend MapItems to be able to present values with both formatted numbers and text.

3.4.5.2.1.25 Legend_Readout_Combo

Legend_Readout_Combo allows a Legend to be defined in a single MapItem as any combination of:

- LEGEND_READOUT
- Popped-up LEGEND_READOUT (This Item is a basic LEGEND_READOUT, but it will appear only when the crew member selects the associated SYMBOL_xxx Item. Disappearance of the popup is implementation dependent)
- Highlighted LEGEND_READOUT (This Item is a basic LEGEND_READOUT, but it will appear only when the associated interactive SYMBOL_x Item is highlighted)

This allows fewer maps items to be used to define the same behavior. This MapItem works similar to the LegendCombo MapItem.

One Legend_Readout_Combo map item with LegendTypes is set to:

Table 3.4.5.2.1.25-1 – Legend_Readout_Combo

LegendType	Equivalent to
A661_LEGEND_ONLY	One A661_LEGEND_READOUT item
A661_POPUP_ONLY	One popped-up A661_LEGEND_READOUT item
A661_HIGHLIGHT_ONLY	One highlighted A661_LEGEND_READOUT item
A661_LEGEND_AND_POPUP	One A661_LEGEND_READOUT item One popped-up A661_LEGEND_READOUT item With the same value
A661_LEGEND_AND_HIGHLIGHT	One A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value
A661_POPUP_AND_HIGHLIGHT	One popped-up A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value
A661_LEGEND_AND_POPUP_AND_HIGHLIGHT	One A661_LEGEND_READOUT item One popped-up A661_LEGEND_READOUT item One highlighted A661_LEGEND_READOUT item With the same value

Legend_Readout_Combo is defined in Table 3.3.22.2.1.25-2

Table 3.4.5.2.1.25-2 – Legend_Readout_Combo

Name	Type	Size (bits)	Description and Value/Range
ItemIndex	ushort	16	
ItemType	uchar	8	A661_LEGEND_READOUT_COMBO
EndFlag	uchar	8	A661_TRUE A661_FALSE
LegendTypes	uchar	8	A661_LEGEND_ONLY A661_POPUP_ONLY A661_HIGHLIGHT_ONLY A661_LEGEND_AND_POPUP A661_LEGEND_AND_HIGHLIGHT A661_POPUP_AND_HIGHLIGHT A661_LEGEND_AND_POPUP_AND_HIGHLIGHT Value of 0 is reserved
UnusedPad	N/A	24	
ReadoutValue	float	32	

3.4.5.2.2 A661_ParameterStructure_BufferOfItems

A66l_ParameterStructure_BufferOfItems as used for MapVert_ItemList is defined in Table 3.4.5.2.2.

Table 3.4.5.2.2 – A661_ParameterStructure_BufferOfItems (Vert)

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_BUFFER_OF_MAPVERT_ITEMS
ClearFlag	1	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
NumberofItems	15	Number of Items modified by the command
{ItemStructures}+	{32}+	

3.4.5.2.3 A661_ParameterStructure_BlockedBufferOfItems

The BlockedBufferOfItems parameter only addresses update synchronization at the MapVert_ItemList widget level. The collection of blocks is treated as a single atomic set. For example, if a buffer comprised of two blocks contains map items 1-200 in block 1, and 201-300 in block 2, all 300 map items will be drawn at the same time.

Update synchronization across multiple map item list widgets, or across multiple layers containing map widgets, is not addressed by this parameter.

A661_ParameterStructure_BlockedBufferOfItems as used for MapVert_ItemList is defined in Table 3.4.5.2.3.

Table 3.4.5.2.3 – A661_ParameterStructure_BlockedBufferOfItems

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_BLOCKED_BUFFER_OF_MAPVERT_ITEMS
ClearFlag	1	If Set, All Items will be set to NOT_USED by CDS before setting the specified Items.
NumberofItems	15	Number of Items modified by the command
LastBlock	8	Indicates the end of the blocked buffer command. A661_FALSE A661_TRUE
UpdateNumber	8	Unique number to indicate the sequence to which this block belongs.
BlockNumber	8	Number indicating the block's position in the sequence.
UnusedPad	8	
{ItemStructures}+	{32}+	

3.4.5.3 MapVert_ItemList Interactive Map Items

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

Overlapping Active Areas:

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

Recommended Behavior For Segment Highlighting and Sending of Events:

Same as for MapHorz_ItemList interactive map items. See Section 3.3.22.3.

3.4.6 EditBoxMultiLine

Categories:

- Graphical representation
- Interactive
- Text String

Description:
EditBoxMultiLine is a text edit box for displaying text across several lines in a scrolling area. The text string can be modified by the crew. The purpose of this widget is to allow free text edit and perform automatic line feed and scroll management.

When the EditBoxMultiLine is in edit mode, the CDS may report all modifications made to the value of the edited string, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the “ReportAllChanges” parameter.

Restriction:
None

EditBoxMultiLine Parameters are defined in Table 3.4.6-1.

Table 3.4.6-1 – EditBoxMultiLine Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_MULTI_LINE
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 interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)

Parameters	Change	Description
		A661_EDB_OPEN_CLOSE CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
LabelString	DR	Text of the edit box
Alignment	D**R**	Justification of the label text within the edit box area CENTER LEFT RIGHT
VerticalScroll	D**R**	Position of scroll controls, absent/left/right/bottom/top
StartCursorPos	DR	Start position of the cursor in field when entering edit **mode. See Section 3.1.3.3.**

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_MULTI_LINE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
StartCursorPos	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
			A661_RIGHT
VerticalScroll	uchar	8	A661_TOP A661_BOTTOM A661_LEFT A661_RIGHT A661_ABSENT
LabelString	string	{8}+	Followed by zero, one, two, or three extra NULL for alignment on 32 bits.

The specific event sent by the EditBoxMultiLine to the owner application is defined in Tables 3.4.6-3, 3.4.6-4, 3.4.6-5, and 3.4.6-6.

Table 3.4.6-3 – EditBoxMultiLine Event Structures: A661_EVT_STRING_CHANGE_ABORTED

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

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

Table 3.4.6-4 – EditBoxMultiLine Event Structures: A661_EVT_STRING_CHANGE

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

Table 3.4.6-5 – EditBoxMultiLine Event Structure: A661_EVT_STRING_CONFIRMED

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

Table 3.4.6-6 – EditBoxMultiLine Event Structures: A661_EVT_EDITBOX_OPENED

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

EditBoxMultiLine Runtime Modifiable Parameters is defined in Table 3.4.6-7.

Table 3.4.6-7 – EditBoxMultiLine Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
StartCursorPos	ushort	16	A661_CURSOR_POS
StyleSet	ushort	16	A661_STYLE_SET
LabelString	string	{32}+	A661_STRING
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
VerticalScroll	**uchar**	**8**	**A661_VERTICAL_SCROLL**

3.4.7 ComboBoxEdit

Categories:

- Graphical representation
- Interactive
- Text String

Description:
Like ComboBox, ComboBoxEdit provides a means to choose one entry within a list by interacting with that entry. This widget is composed of a static part displaying the selected item and a popup 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 interaction.

Moreover, ComboBoxEdit allows the crew to enter new data to the static part of the widget. When ComboBoxEdit is in edit mode, the CDS may report all modification made to the edited string, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the "ReportAllChanges" parameter.

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.

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 interactive
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	D**R**	The X position of the widget reference point
PosY	D**R**	The Y position of the widget reference point
SizeX	D**R**	The X dimension size (width) of the widget (in the closed mode)

Parameters	Change	Description
SizeY	DR	The Y dimension size (height) of the ComboBoxEdit (in the closed mode)
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaWidth	DR	X Size of the area available to display the popup part (the entry list)
SelectingAreaHeight	DR	Y Size of the area available to display the popup part (the entry list)
OpeningMode	DR	Position of the popup part relative to the static part when it is opened: UP CENTERED DOWN
MaxStringLength	D	**Maximum size in bytes (including the NULL terminator) of the label text and of each string array entry.**
Alignment	DR	Justification of the label text within the edit area CENTER LEFT RIGHT
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
StartCursorPos	DR	Start position of the cursor in field when entering edit **mode. See Section 3.1.3.3.**
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be less than or equal to MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list from 1 to NumberOfEntries if an entry is selected and 0 else
OpeningEntry	DR	Entry number which is ensured to be visible when the ComboBoxEdit is opened.

Parameters	Change	Description
		Opening entry is in the range [0; NumberOfEntries] OpeningEntry will be set to 0, if not used.
LabelString	N/A	Text of the new entry entered by the crewmember
EntryList	DR	String array holding the list of entries.

Note: D is Design time. R is Run time.

N/A means that this parameter is only used as an event value.
It is never set by the UA (not at definition time nor at runtime).

ComboBoxEdit Creation Structure is defined in Table 3.4.7-2.

Table 3.4.7-2 – ComboBoxEdit Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	A661_COMBO_BOX_EDIT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
OpeningEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
StartCursorPos	ushort	16	
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
EntryList [NumberOfEntries]	{string}+	8 * string length + PAD	There are “NumberOfEntries” strings. 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_SEL_ENTRY_CHANGE

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

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.

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

Table 3.4.7-6 – ComboBoxEdit Event Structures: A661_EVT_STRING_CONFIRMED

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

Table 3.4.7-7 – ComboBoxEdit Event Structures: A661_EVT_EDITBOX_OPENED

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

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.4.7-8 – ComboBoxEdit Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
StyleSet	ushort	16	A661_STYLE_SET
StartCursorPos	ushort	16	A661_CURSOR_POS
EntryList [up to MaxNumberOfEntries]	{string}+	{32}+	A661_STRING_ARRAY
OpeningEntry	ushort	16	A661_OPENING_ENTRY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	uchar	8	A661_ALIGNMENT
OpeningMode	uchar	8	A661_OPENING_MODE
SelectingAreaWidth	ulong	32	A661_SELECTING_WIDTH
SelectingAreaHeight	ulong	32	A661_SELECTING_HEIGHT

3.4.8 MenuBar

Categories:

- Container

Description:
A MenuBar is a widget containing button-type widgets. Via MenuBar the group of buttons can have their sizes and positions managed by the CDS in a consistent manner. The way MenuBar controls the group and moves the cursor among the buttons is implementation dependent.

COMMENTARY

Example behavior of MenuBar:

When a PopUpMenu attached to a button belonging to a MenuBar is visible, a validation through the cursor on another button of the MenuBar closes the PopUpMenu and activates the newly-selected button. Right/Left arrow keys can also be used to move the focus from one button to another button of the MenuBar.

The buttons contained in the MenuBar are individually defined with the MenuBar as the parent widget. The positions of buttons inside the MenuBar are defined by their own PosX and PosY parameters according to the following rules:

- For Horizontal MenuBar, all buttons inside the MenuBar have the same PosY and SizeY defined by the MenuBar parameters ButtonPos and ButtonSize
- For Vertical MenuBar, all buttons inside the MenuBar have the same PosX and SizeX defined by the MenuBar parameters ButtonPos and ButtonSize

Restriction:
None

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 interactive
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
Specific parameters		
Horizontal	DR	True: MenuBar is horizontal False: MenuBar is Vertical
ButtonPos	DR	If Horizontal =True: Value of the button's parameter PosY If Horizontal =False: Value of the button's parameter PosX
ButtonSize	DR	If Horizontal =True: Value of the button's parameter SizeY If Horizontal =False: Value of the button's parameter SizeX

MenuBar Creation Structure is defined in Table 3.4.8-2.

Table 3.4.8-2 – MenuBar Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MENU_BAR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
Horizontal	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0
PosX	long	32	
PosY	long	32	
ButtonPos	long	32	
ButtonSize	ulong	32	

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.4.8-3 – MenuBar Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size(bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
Horizontal	uchar	8	A661_MENU_HORIZONTAL
ButtonPos	long	32	A661_BUTTON_POS
ButtonSize	ulong	32	A661_BUTTON_SIZE

3.5 Widgets Added for Supplement 2

This section was added in Supplement 2. It introduces new widgets to ARINC 661.

3.5.1 MutuallyExclusiveContainer

Categories:

- Container

Description:
The MutuallyExclusiveContainer has no graphical representation. Its purpose is to group children widgets and to provide a means for managing the exclusive display of the children. The operation of the MutuallyExclusiveContainer is very similar to that of the BasicContainer except that only one child of the MutuallyExclusiveContainer can be visible. All of the children of a BasicContainer can be visible.

A UA can make a child visible in the container by setting the VisibleChild parameter. Only the Widget in the container that has the WidgetIdent that matches the VisibleChild will be processed for display. The visibility of the child is controlled by the child's Visible parameter and is not changed by changing the VisibleChild parameter. For a child in the container to be visible the VisibleChild must be set to the child's WidgetIdent and the Visible parameter for the Widget must be True. Normal rules for visibility and enabling of an ancestor widget apply to the MutuallyExclusiveContainer.

The contained widgets are positioned with respect to the PosX and PosY of the MutuallyExclusiveContainer.

There is no interactivity associated with this widget so there are no events associated with this widget. Only the UA controlling the layer can change the VisibleChild.

The following figure shows the difference between the MutuallyExclusiveContainer and the BasicContainer. The bottom picture shows an example application of the MutuallyExclusiveContainer to make one message visible from a collection of 4 messages.

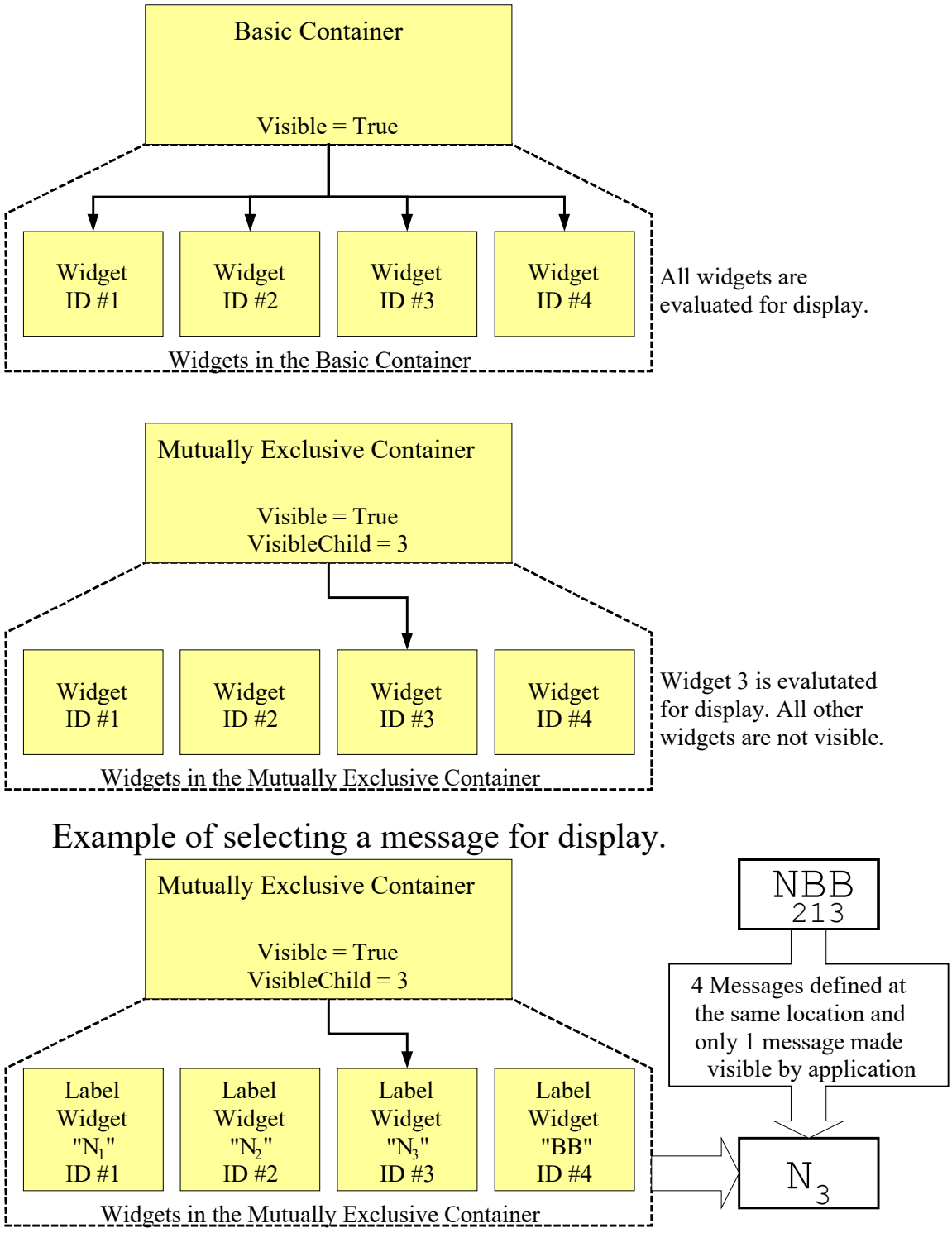


Figure 3.5.1-1 – MutuallyExclusiveContainer

Restriction:
None

MutuallyExclusiveContainer Parameters are defined in Table 3.5.1-1.

Table 3.5.1-1 – MutuallyExclusiveContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MUTUALLY_EXCLUSIVE_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget's descendants to be interactive
Specific parameters		
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
VisibleChild	DR	Identifier of the widget to be made visible. A value of 0 means that no widget is visible

MutuallyExclusiveContainer Creation Structure is defined in Table 3.5.1-2.

Table 3.5.1-2 – MutuallyExclusiveContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_MUTUALLY_EXCLUSIVE_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
VisibleChild	ushort	16	
UnusedPad	N/A	16	0

Mutually Exclusive Container Runtime Modifiable Parameters are defined in Table 3.5.1-3.

Table 3.5.1-3 – MutuallyExclusiveContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
VisibleChild	ushort	16	A661_VISIBLE_CHILD
PosX,PosY	longx 2	32x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y

3.5.2 ProxyButton

Categories:

- Interactive

Description:
The ProxyButton directs a physical button press as a select event to the target widget. If the target widget is visible and enabled at the time of the physical button press, it will respond in the same way as if the user had selected the widget. The CDS will supply a list of unique identifiers for the selection keys available in the system. A common use for this feature is a bezel line select soft key. If the target widget ID is 0, then the select event is sent directly to the UA.

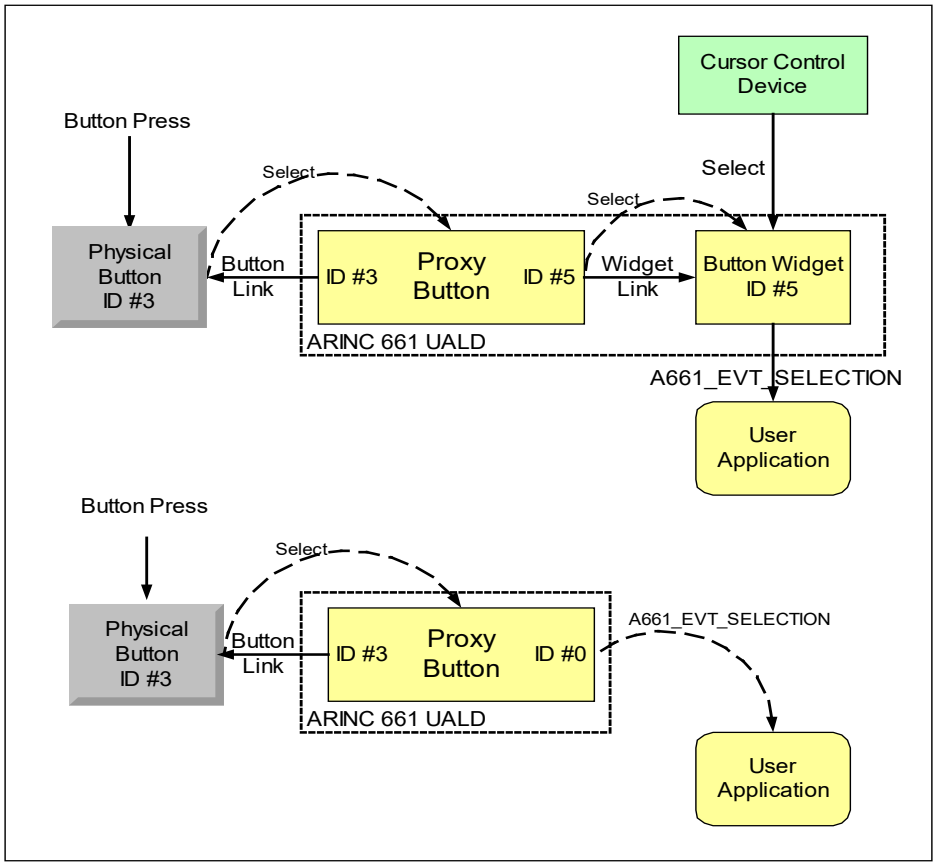


Figure 3.5.2-1 – Proxy Button

Restrictions:

- The result of assigning more than one ProxyButton widget to a single key will cause events to be sent to multiple widgets when that key is pressed
- If an undefined target widget ID is set ,the A661_ERR_SET_ABORTED may be sent to the application
- If the ProxyButton is in a container or layer that is disabled or not visible, then the ProxyButton is disabled. When the ProxyButton is disabled, the ProxyButton will not pass events from physical buttons

ProxyButton Parameters are defined in Table 3.5.2-1.

Table 3.5.2-1 – ProxyButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PROXY_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be interactive
Specific Parameters		
DedicatedKeyIdent	D	The unique identifier of the dedicated CDS key
TargetWidgetIdent	DR	Identifier of the widget to receive the select event

ProxyButton Creation Structure is defined in Table 3.5.2-2.

Table 3.5.2-2 – ProxyButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_PROXY_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
UnusedPad	N/A	8	0
DedicatedKeyIdent	ushort	16	
TargetWidgetIdent	ushort	16	

ProxyButton Event Structures:

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

ProxyButton Event Structures: A661_EVT_SELECTION is defined in Table 3.5.2-3.

Table 3.5.2-3 – ProxyButton Event Structures: A661_EVT_SELECTION

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

Available SetParameter identifiers and associated data structure are defined in Table 3.5.2-4.

Table 3.5.2-4 – ProxyButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
TargetWidgetIdent	ushort	16	A661_TARGET_WIDGET_ID
Enable	uchar	8	A661_ENABLE

3.5.3 WatchdogContainer

Categories:

- Container

Description
The WatchdogContainer widget is a non-graphical container widget stimulated, during normal operation, by the UA at a periodic rate. One or more widgets must be placed into the container and the ShowIfFailIdent must be set to either zero or the value of one of the widgets in the container. During normal operation, all of the widgets within this container, except for the widget referenced with the ShowIfFailIdent, are evaluated for display. If the UA fails to stimulate the Refresh parameter in the Watchdog Container widget for FailCountLimit periods, then the watchdog is considered expired causing an event to be sent back to the UA and causing the CDS to display the widget referenced by ShowIfFailIdent.

This widget is useful when used in combination with the BufferFormat widget to assure that a data set is updated at a specific rate. The BufferFormat contains a set of functionally related parameters and the “Refresh” parameter for a Watchdog widget. If the BufferFormat is updated at the period defined by “TimePeriod,” then the timer is satisfied and no action is taken. If the “Refresh” is not set to true for FailCountLimit periods, the timer expires and two actions are taken by the CDS:

- An event is sent back to the UA indicating that the timer has expired
- The Widget referenced by ShowIfFailIdent is evaluated for display. If the Visible parameter of the child widget is True then the widget is displayed

If the UA starts setting the “Refresh” to true for “ValidCountLimit” periods, the Watchdog is satisfied. When the Watchdog is satisfied the following actions are taken by the CDS:

- An event is sent back to the UA indicating the watchdog is satisfied
- All child widgets of the container except for the widget referenced by ShowIfFailIdent are evaluated for display. If the Visible parameter of a child widget is True, then the widget is displayed.

The timer of a watchdog that is a descendant of another watchdog is still updated regardless of the state of the ancestor watchdog.

In the example below, the first picture shows a buffer format updating the watchdog timer to display widget #12. The Refresh parameter is included in the buffer format. The bottom picture shows the case when the buffer format is no longer received by the CDS. In this case the fail indication in widget #24 is displayed.

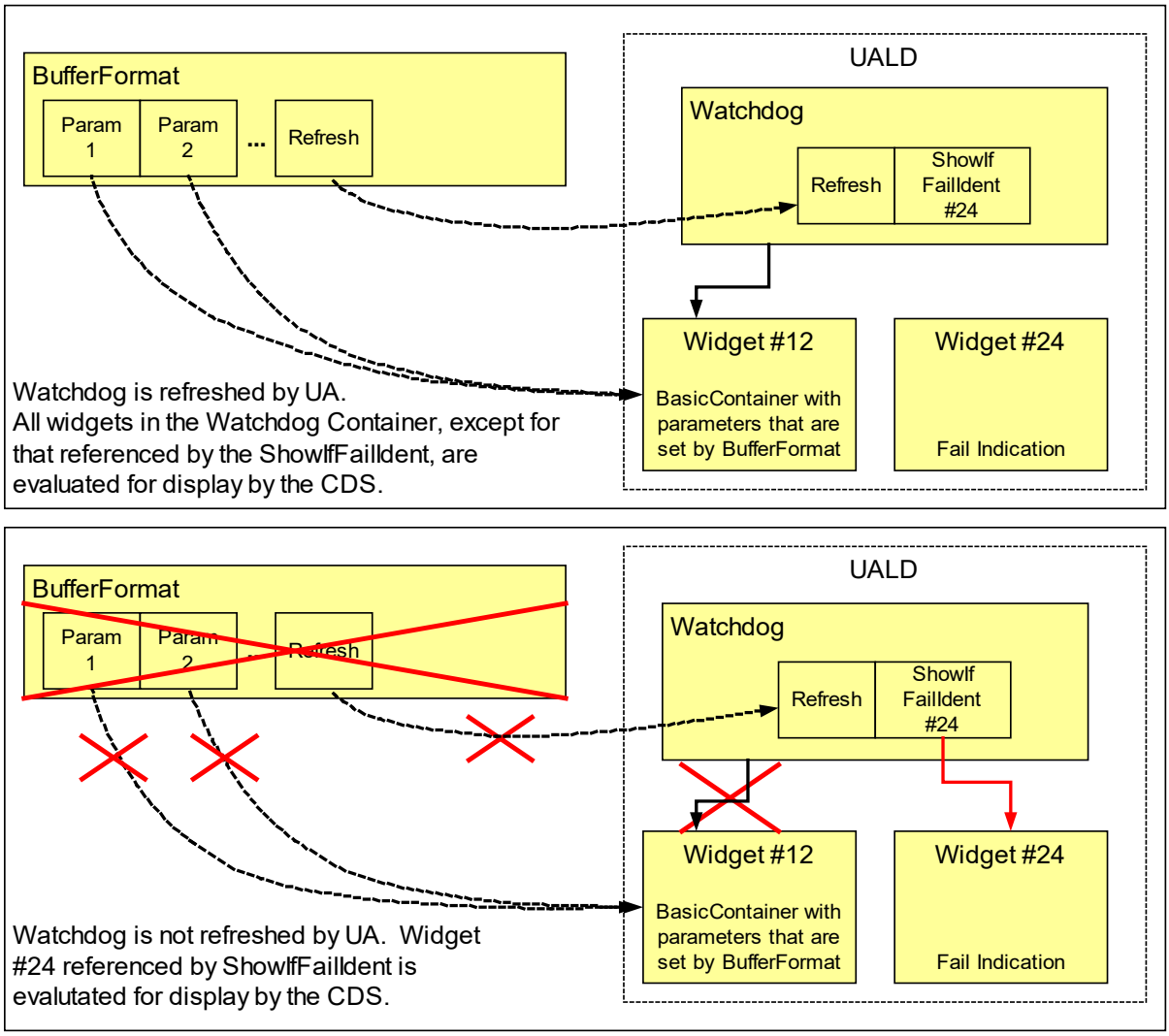


Figure 3.5.3-1 – Watchdog Buffer

To provide a better understanding of how the timer works consider the following example:

- TimePeriod = 50 ms
- ValidCountLimit = 2
- FailCountLimit = 3

In the above example, the UA must set Refresh once every 50 ms for the timer to be satisfied. There are two inner states associated with the watchdog. They are WatchdogExpired and WatchdogNormal. The CDS should evaluate the refresh and timer counts once every 50 ms as follows:

- When the state is WatchdogExpired makes the ShowIfFailIdent widget visible. Every 50 ms (the TimePeriod) the CDS checks to see if the refresh has been updated by the UA in the last 50 ms. If the refresh is updated for 2 consecutive cycles (100ms), then the CDS transitions to the WatchdogNormal state.

- When the state is WatchdogNormal. In the WatchdogNormal state the CDS evaluates the other widgets for display. Every 50 ms (the TimePeriod) the CDS checks to see if Refresh has been updated by the UA in the last 50ms. If the refresh is not updated for 3 consecutive cycles (150 ms), then the CDS transitions to the WatchdogExpired state.

The figure below shows a state transition diagram for the Watchdog Container while the widget is being evaluated. With TimePeriod equal to 50 ms, this State Transition Diagram would be evaluated once every 50 ms by the CDS. For the labels on the transitions the condition above the line determines whether the transition is taken and the statement under the line is the action taken if the transition occurs.

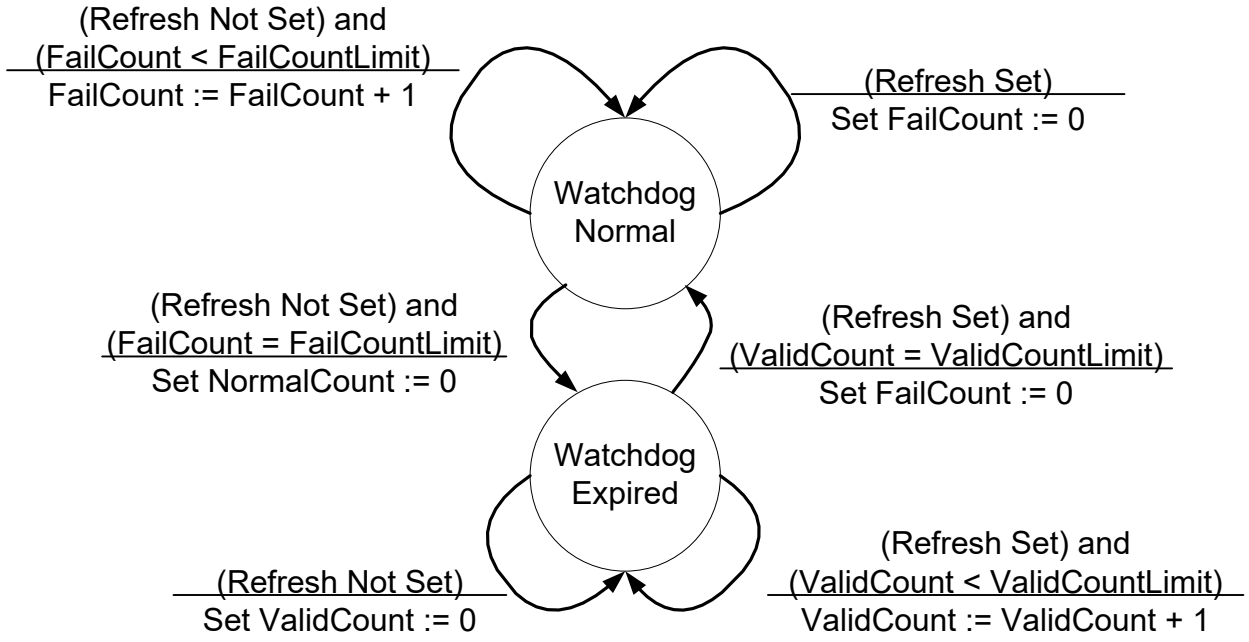


Figure 3.5.3-2 – Watchdog State

The scope of the period during which the widget is evaluating is implementation dependent. The initial state of the state machine is also implementation dependent.

Restriction:
N/A

Table 3.5.3-1 – WatchdogContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_WATCHDOG_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Specific parameters		
TimePeriod	D	The period in ms between updates of timer. Note: The values allowed for this parameter depend on the CDS supplier.
ValidCountLimit	D	The number of periods that the refresh must be updated before the Watchdog is satisfied.
FailCountLimit	D	The number of periods that the refresh is not updated before the Watchdog expires.
Refresh	DR	Set by the UA to satisfy the Watchdog. The significance of the value is implementation dependent.
ShowIfFailIdent	D	The ID of the child widget to be evaluated for display when the timer expires. A value of zero indicates that nothing is to be displayed when the timer expires.
ShowFail	DR	ShowFail can be set to true by the UA to show the failed state of the widget. This has the same result as failing to update the Refresh.

Table 3.5.3-2 – WatchdogContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_WATCHDOG_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	16	0
TimePeriod	ushort	16	
ValidCountLimit	ushort	16	
FailCountLimit	ushort	16	
Refresh	uchar	8	
UnusedPad	N/A	8	0
ShowIfFailIdent	ushort	16	
ShowFail	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	0

Table 3.5.3-3 – WatchdogContainer Event Structures: A661_EVT_WATCHDOG_EXPIRED

CreateParameterBuffer	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_WATCHDOG_EXPIRED
UnusedPad	N/A	16	0

Table 3.5.3-4 – WatchdogContainer Event Structures: A661_EVT_WATCHDOG_NORMAL

CreateParameterBuffer	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_WATCHDOG_NORMAL
UnusedPad	N/A	16	0

Table 3.5.3-5 – WatchdogContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size(bits)	ParameterIdent used in the ParameterStructure
Refresh	uchar	8	A661_REFRESH
ShowFail	uchar	8	A661_SHOW_FAIL

3.5.4 Slider

Categories:

- Graphical Representation
- Interactive

Description:

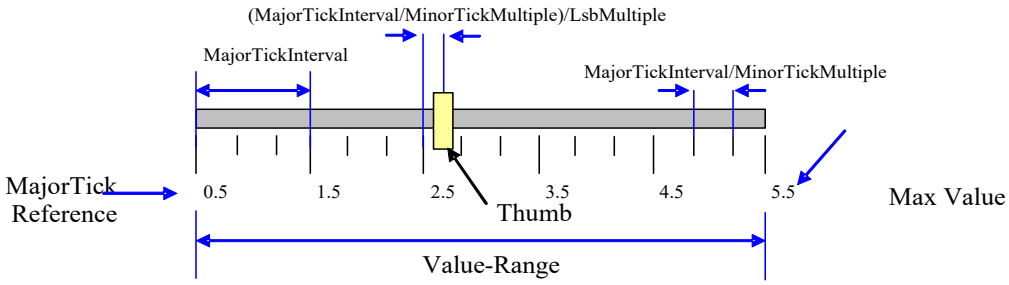


Figure 3.5.4 – Slider

A Slider allows the crewmember to select a value between the range of MIN_VALUE and MAX VALUE. The Slider can be displayed in either the horizontal or vertical axis.

The mapping of the value-range to the slider depends on the orientation, for example when the slider is horizontal the value-range can either be LEFT TO RIGHT or RIGHT TO LEFT. The Orientation specifies direction of increasing values.

When the UA sets the Value to something that is not a multiple of the LSB value the behavior is implementation dependent

The event is reported on upon selection by a crewmember with a “click” or keyboard selection. The CDS would then move the thumb to the click position.

In the diagram above, the following values are used:

- MajorTickReference = 0.5
- MajorTickInterval = 1.0
- MinorTickMultiple = 3
- LsbMultiple = 2

Restriction:
MAX_VALUE must be greater than the MIN_VALUE.

Table 3.5.4-1 – Slider Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SLIDER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MinValue	DR	Minimum value of the scroll bar
MaxValue	DR	Maximum value of the scroll bar
Value	DR	Current value of the Slider
MajorTickInterval	DR	Value between each MAJOR tick mark. MajorTickInterval cannot be zero.
ShowMajorLabels	DR	If TRUE the values for each MAJOR tick are shown.
Alignment	DR	Specifies where the ticks and labels are drawn with relation to the slider. A661_LEFT A661_RIGHT A661_CENTER A661_TOP A661_BOTTOM
Orientation	DR	Specifies the orientation of the Slider either vertical or horizontal. It also specifies which way the value-range of the slider is oriented. A661_BOTTOM_TO_TOP - Vertical A661_TOP_TO_BOTTOM - Vertical A661_LEFT_TO_RIGHT - Horizontal A661_RIGHT_TO_LEFT - Horizontal
LsbMultiple	DR	This specifies the number of intervals between discrete positions for the thumb within the Minor Tick.
MinorTickMultiple	DR	Specifies the number of Minor Tick intervals between two Major Ticks.
MajorTickReference	DR	Holds the value where the major tick starts.
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

Table 3.5.4-2 – Slider Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SLIDER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MinValue	float	32	
MaxValue	float	32	
Value	float	32	
MajorTickInterval	float	32	
ShowMajorLabels	uchar	8	A661_TRUE A661_FALSE
Orientation	uchar	8	A661_BOTTOM_TO_TOP A661_TOP_TO_BOTTOM A661_LEFT_TO_RIGHT A661_RIGHT_TO_LEFT
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER A661_TOP A661_BOTTOM
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
LsbMultiple	ushort	16	
MinorTickMultiple	ushort	16	
MajorTickReference	float	32	

Table 3.5.4-3 – Slider 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	Holds the current value of the Slider

Available SetParameter identifiers and associated data structure are defined in Table 3.5.4-4.

Table 3.5.4-4 – Slider Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
Value	float	32	A661_VALUE
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
MinValue	float	32	A661_MIN_VALUE
MaxValue	float	32	A661_MAX_VALUE
MajorTickInterval	float	32	A661_MAJOR_TICK_INTERVAL
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
MinorTickMultiple	ushort	16	A661_MINOR_TICK_MULTIPLE
MajorTickReference	float	32	A661_MAJOR_TICK_REFERENCE
ShowMajorLabels	uchar	8	A661_SHOW_MAJOR_LABELS
Orientation	uchar	8	A661_ORIENTATION
Alignment	uchar	8	A661_ALIGNMENT
LsbMultiple	ushort	16	A661_LSB_MULTIPLE

3.5.5 PictureAnimated

Categories:

- Graphical representation

Description:
A PictureAnimated is a reference to a set of images available in the CDS. By displaying this set of pictures successively at a frequency defined as a parameter of the widget, the CDS performs an animation. It is possible to define if the animation is performed only once then stopped or if it is an infinite loop. The UA can stop the loop animation. The animation always stops on the first picture of the list.

Restriction:
N/A

Table 3.5.5-1 – PictureAnimated Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PICTURE_ANIMATED
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
IndexOfFrequency	DR	Index of the frequency defined in CDS (implementation dependent)
LoopFlag	DR	Flag for loop animation: FALSE: animation is played once then stops TRUE: loop animation.
AnimationFlag	DR	TRUE: the animation is played FALSE: the animation continues until it reaches the first picture of the array. Then it stops.
NumberOfPictures	D	The number of picture references to store
PictureArray	D	Array of pictures references stored in the CDS. The definition order set the sequence order.

PictureAnimated Creation Structure is defined in Table 3.5.5-2.

Table 3.5.5-2 – PictureAnimated Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PICTURE_ANIMATED
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NumberOfPictures	ushort	16	
IndexOfFrequency	uchar	8	
LoopFlag	uchar	8	A661_FALSE A661_TRUE
AnimationFlag	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
PictureArray[NumberOfPictures]	{ushort}+	16* NumberOfPictures + PAD	Array of pictures references Followed by zero or two extra NULL for alignment on 32 bits.

The PictureAnimated widget does not send any event.

Available SET_PARAMETER identifiers and associated data structure are:

Table 3.5.5-3 – PictureAnimated Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
AnimationFlag	uchar	8	A661_ANIMATION_FLAG
StyleSet	ushort	16	A661_STYLE_SET
IndexOfFrequency	uchar	8	A661_INDEX_OF_FREQUENCY
LoopFlag	uchar	8	A661_LOOP_FLAG

3.5.6 SymbolAnimated

Categories:

- Graphical representation

Description
The SymbolAnimated widget is similar to the Symbol widget. It places vector symbols predefined in the CDS at a specified location on the display. However, the Symbol widget is static unless the controlling application changes widget parameters, while the SymbolAnimated widget is animated by providing the ability to automatically display a sequence of symbol references, rotation angles and relative movements.

NumberOfSymbols specifies how many elements are specified in the creation structure for each of the following arrays: a symbol reference array, a rotation angle array, and arrays for both an x and a y component of relative movement (relative to the widgets overall PosX/PosY position). The combination of these arrays allows the definition of animation sequences in which the symbol displayed by the CDS, its orientation and position relative to the widget's origin can change over the course of time.

IndexOfFrequency is interpreted according to implementation dependent rules. This widget attribute controls the time between adjacent cycles of the animation sequence when the animation is played.

AnimationType allows the controlling application to start and stop the animation. When the widget becomes visible and AnimationType is set to A661_DONT_RUN, the first symbol reference, orientation, and movement parameters are used to display the symbol. If AnimationType is A661_RUN_ONCE or A661_RUN when the widget is set to visible, the animation begins. A661_RUN_ONCE will start the animation and have the CDS stop it automatically once the animation sequence has been completely displayed – to start over, AnimationType must be set to A661_RUN_ONCE again. To show the animation continuously, AnimationType should be set to A661_RUN and remain this value until the animation should stop, at which time the value should be reset to A661_DONT_RUN.

LoopType is one of the following:

- A661_LOOP_FORWARD:
  The sequence is played from the first to the last frame. When the animation is stopped, the last frame remains visible
- A661_LOOP_FORWARD_AND_RESET:
  The sequence is played from the first to the last frame. When the animation is stopped, the first frame remains visible
- A661_LOOP_FORWARD_AND_BLANK:
  While the animation is running, the sequence is played from the first to the last frame. When the animation is not running, nothing is visible
- A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK:
  While the animation is running, the sequence is played from the first to the last frame and then reverses direction until the first frame is reached again. When the animation is not running, nothing is visible

Restriction:
N/A

Table 3.5.6-1 – SymbolAnimated Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SYMBOL_ANIMATED
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside the CDS
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
Specific parameters		
MotionAllowed	D	Capability to change PosX, PosY at run time
ColorIndex	DR	Color index **see Section 3.1.3.3.1**
IndexOfFrequency	DR	Index of the frequency defined in the CDS (implementation dependent)
LoopType	DR	Type of looping (see description above)
AnimationType	DR	Type of animation (see description above)
NumberOfSymbols	D	The number of symbol references used for this animation.
SymbolArray	D	Array of symbol references
OrientationArray	D	Array of symbol orientations
MovementXArray	D	Array of horizontal component of movement
MovementYArray	D	Array of vertical component of movement

Table 3.5.6-2 – SymbolAnimated Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_SYMBOL_ANIMATED
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
MotionAllowed	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
StyleSet	ushort	16	
ColorIndex	uchar	8	
LoopType	uchar	8	A661_LOOP_FORWARD A661_LOOP_FORWARD_AND_RESET A661_LOOP_FORWARD_AND_BLANK A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK
NumberOfSymbols	ushort	16	
IndexOfFrequency	uchar	8	
AnimationType	uchar	8	A661_DONT_RUN A661_RUN A661_RUN_ONCE
SymbolArray	{ushort}+	16 * NumberOf Symbols + PAD	Array of symbol references, padded if necessary, to reach a 32-bit boundary.
OrientationArray	{fr(180)}+	32 * NumberOf Symbols	Array of rotation angles.
MovementXArray	{long}+	32 * NumberOf Symbols	Array of horizontal movements.
MovementYArray	{long}+	32 * NumberOf Symbols	Array of vertical movements.

Table 3.5.6-3 – SymbolAnimated Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
ColorIndex	uchar	8	A661_COLOR_INDEX
AnimationType	uchar	8	A661_ANIMATION_TYPE
IndexOfFrequency	**uchar**	**8**	**A661_INDEX_OF_FREQUENCY**
LoopFlag	**uchar**	**8**	**A661_LOOP_FLAG**

3.5.7 SelectionListButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The SelectionListButton is the same as the ComboBox except that a constant LabelString is displayed instead of the selected value.

The SelectionListButton allows a crew member to select one entry within a list. A fixed LabelString is displayed in the SelectionListButton area. The number of the current selected entry is held in the SelectedEntry parameter. The complete list of possible Entries is held in a string array (parameter EntryList). The list is displayed upon a crew member interaction.

Note that SelectingAreaHeight and the SelectingAreaWidth represent the Y and X Size of the PopUp part of the SelectionListButton.

OpeningMode of the SelectionListButton determines how the SelectionListButton opens.

Restriction:
N/A

SelectionListButton Parameters are defined in Table 3.5.7 -1.

Table 3.5.7-1 – SelectionListButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SELECTION_LIST_BUTTON
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the SelectionListButton (in the closed mode)
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
SelectingAreaWidth	DR	X Size of the area available to display the popup part (the entry list)
SelectingAreaHeight	DR	Y Size of the area available to display the popup part (the entry list)
LabelString	D	Text permanently displayed in the SelectionListButton label area.
OpeningMode	DR	Position of the popup part relative to the static part when it is opened: UP CENTERED DOWN
MaxStringLength	D	**Maximum size in bytes (including the NULL terminator) of each string array entry.**

Parameters	Change	Description
Alignment	DR	Justification of the text within the label area of the widget LEFT RIGHT CENTER
MaxNumberOfEntries	D	Maximum number of entries in the list
NumberOfEntries	DR	Total number of entries in the list (must be less than or equal to MaxNumberOfEntries)
SelectedEntry	DR	Current selected entry number in the list.
OpeningEntry	DR	Entry number which is ensured to be visible when the SelectionListButton is opened. Opening entry is in the range [0; NumberOfEntries] OpeningEntry will be set to 0, if not used.
EntryList	DR	String array holding the list of entries.
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

SelectionListButton Creation Structure is defined in Table 3.5.7-2.

Table 3.5.7-2 – SelectionListButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SELECTION_LIST_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SelectingAreaWidth	ulong	32	
SelectingAreaHeight	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
OpeningEntry	ushort	16	
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
OpeningMode	uchar	8	A661_OPEN_UP A661_OPEN_CENTERED A661_OPEN_DOWN

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0
LabelString	string	8 x string length+ PAD	The string is followed by zero, one, two, or three NULL character(s) to be 32 bits aligned.
EntryList [NumberOfEntries]	{string}+	{32}+	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.

The specific event sent by the SelectionListButton to the owner application is defined by Table 3.5.7-3.

Table 3.5.7-3 – SelectionListButton Event Structures: A661_EVT_SEL_ENTRY_CHANGE

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

Available SetParameter identifiers and associated data structure are defined in Table 3.5.7-4.

SelectionListButton Runtime Modifiable Parameters are defined in Table 3.5.7-4.

Table 3.5.7-4 – SelectionListButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
OpeningEntry	ushort	16	A661_OPENING_ENTRY
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
EntryList [up to MaxNumberOfEntries]	{string}+	{32}+	A661_STRING_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
SelectingAreaWidth	**ulong**	**32**	**A661_SELECTING_WIDTH**
SelectingAreaHeight	**ulong**	**32**	**A661_SELECTING_HEIGHT**
OpeningMode	**uchar**	**8**	**A661_OPENING_MODE**
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.6 Widgets Added for Supplement 3

3.6.1 EditBoxNumericBCD

Categories:

- Graphical representation
- Interactive
- Text string

Description:
The EditBoxNumericBCD is a customized numeric edit box allowing the user to edit a complex numeric value that is not in base 10 (such as a time). The principle of this widget is to use several fields for the value, each field being a part of the value (e.g., for time = 23h59 min, there is one field for hour from 0 to 23 and one field for minutes from 0 to 59). A crew member can modify this value using input devices. As it is a numeric value, CDS is able to increment the value. The widget can receive a number of increments or a numeric key value.

The idea is to use distinct fields within the 64-bit “Value” Parameter. The size of each field is defined in the widget. Each 4-bit part of the “Value” parameter is used to code a digit (in BCD format) of the value to be displayed. The first 4-bit field is used to code the sign (0xF=-1 / 0x1=+1).

Example:

0x**F**123456789012345:

sign: odd (**F**)

1st digit = 1

2nd digit = 2

…

15th digit = 5

The number of fields is defined by the parameter NumberOfFields.

The table SizeOfFields[NumberOfFields] allows you to define the size of each field in digits. Note that the format limits the global size parameter value to 15 digits. It is not necessary to use all 15 digits. It is not allowed to define a field size of 0.

Table 3.6.1-1 – SizeOfFields

	Example 1	**Example 2**	**Example 3**
NumberOfFields	2	4	2
SizeOfFields	3; 2	3; 2; 2; 4	2; 2
Value	0x1123450000000000	0xF179595912340000	0x1094500000000000
decoding	Sign: + Field1: 123 Field2: 45	Sign: - Field1: 179 Field2: 59 Field3: 59 Field 4: 1234	Sign: + Field1: 09 Field2: 45

The widget parameters MinFieldsValue and MaxFieldsValue define the min and max boundaries of each field in ABSOLUTE VALUE (always positive). The widget parameters MinFieldsValues and MaxFieldsValues use the same coding of ‘value’ parameter but the sign is not interpreted.

MinValue and MaxValue parameters allow you to define the minimum and maximum of the global signed value.

MinValue and MaxValue parameters use the same coding as the Value parameter.

Table 3.6.1-2 – MinFieldsValues and MaxFieldsValues

	Example 1	Example 2
NumberOfFields	4	2
SizeOfFields	3; 2; 2; 4	2; 2
MinFieldsValues	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 Field2: 0 => 0x000000000000000
MaxFieldsValues	Field1: 180 Field2: 59 Field3: 59 Field 4: 9999 => 0x0180595999990000	Field1: 23 Field2: 59 => 0x0235900000000000
MinValue	0xF179595999990000 - 17959599999	0x1000000000000000 0
MaxValue	0x1180000000000000 + 18000000000	0x1235900000000000 + 2359

The widget parameters TicsFine and TicsCoarse use the same coding as the Value parameter.

The parameters PositiveString and NegativeString define a string for positive and negative values. The character '+' in parameter FormatString indicates the position of parameter PositiveString or NegativeString for display.

Table 3.6.1-3 – PositiveString and NegativeString Definitions

	Example 1	Example 2	Example 3
NumberOfFields	4	4	2
SizeOfFields	3; 2; 2; 4	3; 2; 2; 4	2; 2
PositiveString	EAST	null	RIGHT
NegativeString	WEST	null	LEFT
FormatString	###°##'##.####"+	+__#°##'##.#___"	##.## +
MinFieldsValue	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 Field2: 0 Field3: 0 Field4: 0 => 0x0000000000000000	Field1: 0 = 0x00 Field2: 0 = 0x00 => 0x0000000000000000
MaxFieldsValue	Field1: 180 Field2: 59 Field3: 59 Field4: 9999 => 0x0180595999990000	Field1: 360 Field2: 59 Field3: 59 Field4: 9999 => 0x0360595999990000	Field1: 90 Field2: 99 => 0x0909900000000000
MinValue	-179 59 59 9999 => 0xF179595999990000	0 00 00 0000 => 0x1000000000000000	-90 00 => -9000 => 0xF900000000000000
MaxValue	180 00 00 0000 => 18000000000 => 0x1180000000000000	359 59 59 9999 => 35959599999 => 0x1359595999990000	90 00 => 9000 => 0x1900000000000000

	Example 1	Example 2	Example 3
TicsFine	Field1: 0 Field2: 0 Field3: 0 Field4: 50 => 0x0000000000500000	Field1: 0 Field2: 0 Field3: 0 Field4: 50 => 0x0000000000500000	Field1: 0 Field2: 1 => 0x0000100000000000
TicsCoarse	Field1: 0 Field2: 0 Field3: 0 Field4: 1000 => 0x0000000010000000	Field1: 0 Field2: 0 Field3: 0 Field4: 500 => 0x0000000005000000	Field1: 1 Field2: 0 => 0x0010000000000000
Value (example)	Sign: + Field1: 45 Field2: 5 Field3: 10 Field4: 4200 0x1045051042000000	Sign: + Field1: 45 Field2: 5 Field3: 10 Field4: 4200 0x1045051042000000	Sign: - Field1: 79 Field2: 79 0xF797900000000000
Display	045°05'10.4200"EAST	45°05'10.42"	79.79 LEFT

Some of the parameters of the EditBoxNumericBCD have to be consistent:

- Each field of the Value must belong to [MinFieldValue, MaxFieldValue]
- The Value must belong to [MinValue, MaxValue] and be consistent with FormatString
- MinValue must be less than MaxValue
- MinFieldValue must be less than MaxFieldValue
- The length of the string being edited and the length of the FormatString must be less than MaxFormatStringLength
- During the run-time phase, if the UA sets any field of parameter “Value” with a value inferior to MinFieldValue or greater than MaxFieldValue, the CDS reports an error. In the same way, if during the run-time phase, the UA sets the parameter Value with a value inferior to MinValue or greater than MaxValue, the CDS reports an error. However, the values entered by the crew member are not checked by the CDS.

When the EditBoxNumericBCD is in Edit Mode, the CDS may report all modifications made to the edited value, some of the modifications, or just the confirmed value (after a crew member validation). The UA controls this option through the “ReportAllChanges” parameter.

Upon event notification A661_STRING_CHANGE & A661_STRING_CONFIRMED (respectively A661_STRING_CHANGE_ABORTED), the string value returned to the UA is the string currently displayed in the edit area (respectively the ‘value’ parameter converted in string according to formatString).

EditBoxNumericBCD Parameters are defined in Table 3.6.1-4.

Table 3.6.1-4 – EditBoxNumericBCD Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_EDIT_BOX_NUMERIC_BCD
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter.
Specific parameters		
Value	DR	Value to be interpreted (format Binary-Coded Decimal). Each half-byte will be interpreted as a digit to be displayed in the final string except the most significant one which will be used for the sign (F for a negative value and 1 for a positive value). Example: 0xF123456789012345 - negative sign (F) - 1st digit = 1 - 2nd digit = 2 - … - 15th digit = 5
NumberOfFields	D	Number Of Fields in the Value
SizeOfFields [NumberOfFields]	D	Number of digits of each field of the Value
MinFieldsValues	D	Minimum value of each field (in absolute value)
MaxFieldsValues	D	Maximum value of each field (in absolute value)
MinValue	D	Minimum value of the parameter Value
MaxValue	D	Maximum value of the parameter Value
TicsCoarse	DR	Coarse increment step for modification of the value with main wheel

<table>
<tr><th>Parameters</th><th>Change</th><th>Description</th></tr>
<tr><td>FormatString</td><td>DR</td><td>String describing the format of the numeric field. The string is composed of any authorized characters. Some of them will be interpreted to format the value.
‘_’ and ‘#’ are used to position the different digits within the string (the first ‘_’ or ‘#’ indicates the position of the first digit and so on …).

‘_’ : is used to remove leading and trailing 0 within the same field. If the digit is equal to 0, it indicates that the corresponding digit is replaced by no character, or it is displayed when there is a digit on the left and on the right of the 0 within the same field. In other cases ([1-9]), it indicates that the corresponding digit will be displayed as it is defined in the value. Refer to Figure 3.6.1-2 for more information.

‘#’ : indicates that the corresponding digit will be displayed as it is defined in the value.
‘+’: defines the position of the PositiveString/NegativeString in the final string. Note that if this sign character is not present in the FormatString, there will be no difference between the display of a positive or a negative value.
‘*’: Any other characters (any except one of ‘+’, ‘_’, ‘#’) will be displayed

Examples:

<table>
<tr><td>NumberOfFields/ SizeOfFields</td><td>PosString/NegString FormatString</td><td>Value</td><td>Displayed string</td></tr>
<tr><td>2 / 4; 1</td><td>+ / - +___#.#</td><td>0xF0123400</td><td>- 123.4</td></tr>
<tr><td>4 / 3; 2; 2; 4</td><td>EAST/WEST ###°##'##.###"+</td><td>0xF179595913240000</td><td>179°59'59.1234 WEST</td></tr>
<tr><td>3 / 3; 2; 2</td><td>null / - +###°##'##"</td><td>0x10054630</td><td>005°46'30"</td></tr>
<tr><td>2 / 2; 2</td><td>RIGHT/LEFT _#.#_ +</td><td>0x10510000</td><td>5.1 RIGHT</td></tr>
<tr><td>2 / 3; 2</td><td>+ / - ###x##</td><td>0x12501500</td><td>250x15</td></tr>
</table>
</td></tr>
<tr><td>MaxFormatStringLength</td><td>D</td><td>Maximum size in bytes (including the NULL terminator) of FormatString.</td></tr>
<tr><td>PositiveStringLength</td><td>D</td><td>Maximum size in bytes (including the NULL terminator) of PositiveString.</td></tr>
<tr><td>NegativeStringLength</td><td>D</td><td>Maximum size in bytes (including the NULL terminator) of NegativeString.</td></tr>
<tr><td>PositiveString</td><td>DR</td><td>String to be used for a positive value</td></tr>
<tr><td>NegativeString</td><td>DR</td><td>String to be used for a negative value</td></tr>
<tr><td>Alignment</td><td>DR</td><td>Justification of the label text within the edit box area
CENTER
LEFT
RIGHT</td></tr>
<tr><td>TicsFine</td><td>DR</td><td>Fine increment step for modification of the value with secondary wheel</td></tr>
</table>

Parameters	Change	Description
ReportAllChanges	D	**A661_EDB_CHANGE_CONFIRMED** CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) **A661_EDB_ALL_CHANGE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report each update from the crew member while in edit mode (A661_EVT_STRING_CHANGE) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED) **A661_EDB_OPEN_CLOSE** CDS will report the edit mode opening (A661_EVT_EDITBOX_OPENED) CDS will report the value change after crew member validation (A661_EVT_STRING_CONFIRMED) CDS will report the value if the crew member aborts the edit (A661_EVT_STRING_CHANGE_ABORTED)
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE
MaxLegendStringLength	D	Max string size for the legend (MaxLegendStringLength > 0)
LegendAreaSizeX	DR	The X dimension (size) of the legend.
LegendString	DR	Legend associated to the numeric value
LegendPosition	DR	Position of the legend in comparison with the numeric value: LEFT, RIGHT, TOP, BOTTOM
LegendRemoved	DR	The flag defining if the legend is removed on entry in the editing mode.
NumericKeyFlag	D	Ability to change the value with the numerical key TRUE FALSE
CyclicFlag	D	Possibility for cyclic modification of the value TRUE FALSE

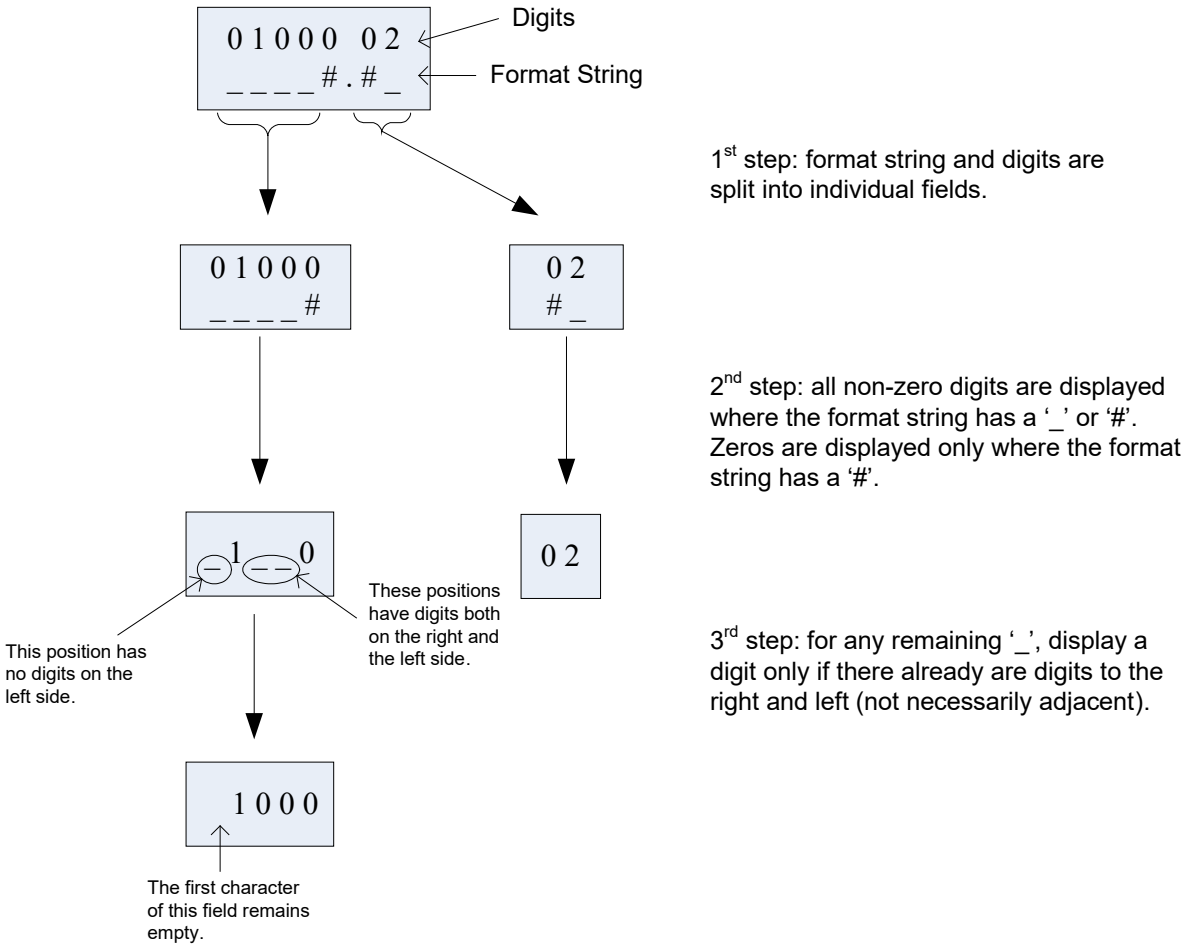


Figure 3.6.1-1 – Format String Example of '_' Character

EditBoxNumeric Creation Structure is defined in Table 3.6.1-5.

Table 3.6.1-5 – EditBoxNumericBCD Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range when necessary
WidgetType	ushort	16	A661_EDIT_BOX_NUMERIC_BCD
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
LegendAreaSizeX	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
Value	BCD	64	Comment: Each 4 bits will be interpreted as a digit to be displayed in the final string except the most significant one which will be used for the sign (F for a negative value and 1 for a positive value).
MinValue	BCD	64	Comment: Same type as value
MaxValue	BCD	64	Comment: Same type as value
MinFieldsValues	BCD	64	Comment: Same type as value but the sign will not be interpreted.
MaxFieldsValues	BCD	64	Comment: Same type as value but the sign will not be interpreted.
TicsCoarse	BCD	64	Comment: Same type as value
TicsFine	BCD	64	Comment: Same type as value
NumberOfFields	uchar	8	
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
LegendPosition	uchar	8	A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
LegendRemoved	uchar	8	A661_FALSE A661_TRUE
MaxFormatStringLength	uchar	8	
MaxLegendStringLength	uchar	8	
PositiveStringLength	uchar	8	
NegativeStringLength	uchar	8	

CreateParameterBuffer	Type	Size (bits)	Value/Range when necessary
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
ReportAllChanges	uchar	8	A661_EDB_CHANGE_CONFIRMED A661_EDB_ALL_CHANGE A661_EDB_OPEN_CLOSE
NumericKeyFlag	uchar	8	A661_FALSE A661_TRUE
CyclicFlag	uchar	8	A661_FALSE A661_TRUE
SizeOfFields [NumberOfFields]	uchar	8 * NbOfFields + PAD	Followed by zero, one, two, or three NULL for alignment on 32 bits
FormatString	string	8 * string length	
LegendString	string	8 * string length	
PositiveString	string	8 * string length	
NegativeString	string	8 * string length + PAD	Followed by zero, one, two, or three extra NULL for alignment on 32 bits.

During pilot entry of an EditBoxNumericBCD, it is possible for the pilot to enter values containing alphanumeric character (for example, the value and the legend). It is the responsibility of the UA to analyze the entered value and format it correctly in the widgets. As a consequence of this need, the events associated to the EditBoxNumericBCD take a string value as parameters.

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE_ABORTED are defined in Table 3.6.1-6.

Table 3.6.1-6 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE_ABORTED

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

The CDS should place the most recently validated LabelString in the String parameter field when sending this event to the User Application.

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE are defined in Table 3.6.1-7.

Table 3.6.1-7 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CHANGE

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

EditBoxNumericBCD Event Structures: A661_EVT_STRING_CONFIRMED are defined in Table 3.6.1-8.

Table 3.6.1-8 – EditBoxNumericBCD Event Structures: A661_EVT_STRING_CONFIRMED

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

EditBoxNumericBCD Event Structures: A661_EVT_EDITBOX_OPENED are defined in Table 3.6.1-9.

Table 3.6.1-9 – EditBoxNumericBCD Event Structures: A661_EVT_EDITBOX_OPENED

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

The CDS sends an event to the User Application when the EditBox widget goes to Editing state. As a consequence, the CDS also notifies the UA when the EditBox widget exits the Editing State without any validation, using the event A661_STRING_CHANGE_ABORTED.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.1-10.

Table 3.6.1-10 – EditBoxNumericBCD Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
EntryValidation	uchar	8	A661_ENTRY_VALID
StyleSet	ushort	16	A661_STYLE_SET
Value	BCD	64	A661_VALUE_8BYTE
TicsCoarse	BCD	64	A661_TICS_COARSE_8BYTE
TicsFine	BCD	64	A661_TICS_FINE_8BYTE
LegendString	string	{32}+	A661_STRING
FormatString	string	{32}+	A661_FORMAT_STRING
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
PositiveString	**string**	**{32}+**	**A661_POSITIVE_STRING**
NegativeString	**string**	**{32}+**	**A661_NEGATIVE_STRING**
LegendAreaSizeX	**ulong**	**32**	**A661_LEGEND_AREA_SIZE_X**
LegendPosition	**uchar**	**8**	**A661_LEGEND_POSITION**
LegendRemoved	**uchar**	**8**	**A661_LEGEND_REMOVED**

3.6.2 CursorRef

Categories:

- Utility

Description:
The CursorRef widget defines a non-visible reference point that the Cursor can be moved to using the A661_REQ_CURSOR_ON_WIDGET command.

The ability for a UA to request that the cursor be placed on an item in its layer is important for certain Human Engineering considerations. These include situations where it is important to place the cursor so that it is not on a widget (i.e., at a point on the screen), to avoid inadvertent selection, or in cases where the widget that the cursor is being placed upon is complex and the cursor needs to be placed on a particular sub-part of that widget.

The widget has no graphics associated with it. The values of PosX, PosY defines its location.

The CursorRef can be the normal child of a container, layer, MapHorz_Source or MapVert_Source. Depending upon this parent, the widget parameters PosX and PosY are interpreted as either screen coordinates (1/100mm) or the coordinates of the associated map source. Values of PosX, PosY must be set appropriately to this coordinate system for the CursorRef to be positioned at the intended point.

Restriction:
None

CursorRef Parameters are defined in Table 3.6.2-1.

Table 3.6.2-1-1 – CursorRef Parameters

Parameters	Change	Description
WidgetType	D	A661_CURSOR_REF
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
PosX	DR	The X/Lat/Range position of the widget reference point. First coordinate of reference point
PosY	DR	The Y/Long/Bearing position of the widget reference point. Second coordinate of reference point

CursorRef Creation Structure is defined in Table 3.6.2-2.

Table 3.6.2-2 – CursorRef Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CURSOR_REF
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	16	0
PosX	long/ scaled Integer	32	If parent is a MapHorz_Source then the LSB and units are defined by the MapHorz_Source. If parent is not a MapHorz_Source then the units are 1/100mm.
PosY	long/ scaled integer	32	If parent is a MapHorz_Source then the LSB and units are defined by the MapHorz_Source. If parent is not a MapHorz_Source then the units are 1/100mm.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.2-3.

Table 3.6.2-3 – CursorRef Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
PosX	Long/scaled integer	32	A661_POS_X
PosY	Long/scaled integer	32	A661_POS_Y
PosX, PosY	long x 2 / scaled integer x 2	32 x 2	A661_POS_XY

3.6.3 CursorOver

Categories:

- Interactive
- Graphical Representation

Description:
The CursorOver widget allows the UA to be notified as soon as a Cursor enters or exits the active area of the widget. The events generated do not require a cursor select action, they are generated when the X,Y position of the cursor intersects with the active area of the widget.

Unlike the ActiveArea and CursorPosOverlays widgets, no additional events are created by the CursorOver widget when the widget is clicked-on.

The events that are generated when the cursor enters the active area of the widget can be configured as follows, using the PositionReport Mode parameter.

- Report All: Events are sent when the cursor enters, exits, or while over the active area.
- On Transition: Events are sent only when the cursor enters or exits the active area.

The event notification reports the X,Y position of the Cursor with respect to the origin of the CursorOver widget.

The StyleSet may be used to define alternate cursor representations.

While the Cursor is over the widget, an alternative CDS defined cursor or other graphics can be displayed if desired, as defined by the widget StyleSet.

The EventOrigin parameter contained in the A661_NOTIFY_WIDGET_EVENT block defines which cursor has entered the active area (see Section 4.5.4.2).

CursorOver widgets that are drawn on top of other interactive widget do not prevent the widget underneath from generating events. Under these conditions the CursorOver widget and the interactive widget underneath will still be able to generate events. This is one of the noted exceptions to the recommendation defined in Section 2.3.5.2 for overlapping widgets.

There are many possible uses for this widget for detecting whether a cursor is within a certain region on the screen. Some example cases are shown below.

Example case 1: By configuring the widget to report on transition, events are only sent to the UA when the cursor passes outside to inside and inside to outside.

Typical Use - Case 1 (On Transition)

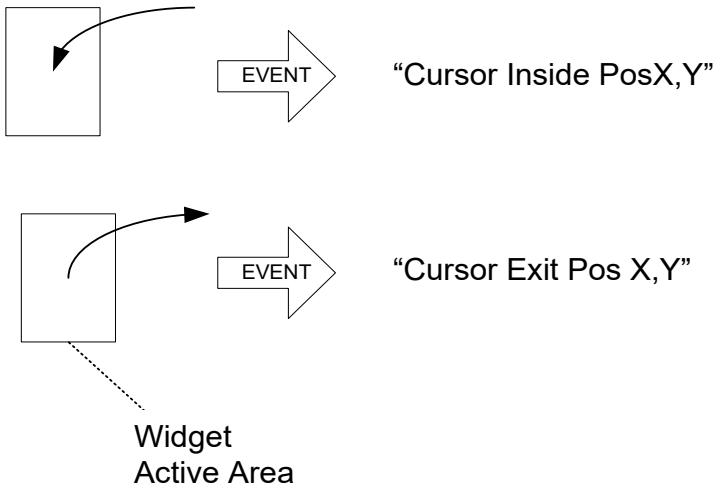


Figure 3.6.3-1 – Typical Use – Case 1 (On Transition)

Example case 2: By configuring the widget to report all, events are sent to the UA when the cursor passes from outside to inside, continuously while the cursor is inside, and from inside to outside.

Typical Use - Case 2 (Report All)

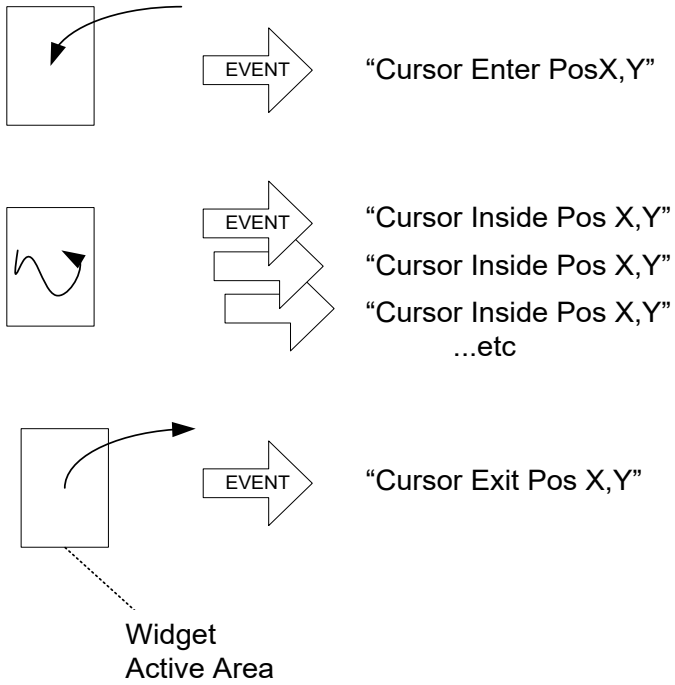


Figure 3.6.3-2 – Typical Use – Case 2 (Report All)

Restriction:
None

CursorOver Parameters are defined in Table 3.6.3-1.

Table 3.6.3-1 – CursorOver Parameters

Parameters	Change	Description
WidgetType	D	A661_CURSOR_OVER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be interactive
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The width of the widget.
SizeY	DR	The height of the widget.
PositionReportMode	D	Defines when events are generated from the widget, when the Cursor is over the widget active area. A661_REPORT_ON_TRANSITION - A661_EVT_CURSOR_ENTER, sent once when cursor enters the active area. - A661_EVT_CURSOR_EXIT, sent once when cursor exits the active area. A661_REPORT_ALL - A661_EVT_CURSOR_ENTER, sent once when cursor enters the active area. - A661_EVT_CURSOR_EXIT, sent once when cursor exits the active area. - A661_EVT_CURSOR_INSIDE, sent continuously while the cursor remains inside the active area.
StyleSet	DR	Defines optional characteristics of the cursor or other related graphics as defined by the CDS implementation.

CursorOver Creation Structure is defined in Table 3.6.3-2.

Table 3.6.3-2 – CursorOver Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_CURSOR_OVER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0
PositionReportMode	uchar	8	A661_REPORT_ON_TRANSITION A661_REPORT_ALL
UnusedPad	N/A	24	0

Note: The Visible and Enable parameters are in a different order to the normal convention.

The specific events sent by the CursorOver to the owner application are defined in Tables 3.6.3-3, -4, and -5.

Table 3.6.3-3 – CursorOver Event Structures: A661_EVT_CURSOR_ENTER

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_ENTER
UnusedPad	N/A	16	0
RelPosX	long	32	Relative X position of Cursor as an offset to the widget's origin.
RelPosY	long	32	Relative Y position of Cursor as an offset to the widget's origin.

Table 3.6.3-4 – CursorOver Event Structures: A661_EVT_CURSOR_INSIDE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_INSIDE
UnusedPad	N/A	16	0
RelPosX	long	32	Relative X position of Cursor as an offset to the widget's origin.
RelPosY	long	32	Relative Y position of Cursor as an offset to the widget's origin.

Table 3.6.3-5 – CursorOver Event Structures: A661_EVT_CURSOR_EXIT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_EXIT
UnusedPad	N/A	16	0
RelPosX	long	32	Relative X position of Cursor as an offset to the widget's origin.
RelPosY	long	32	Relative Y position of Cursor as an offset to the widget's origin.

Available SetParameter identifiers and associated data structure are defined in Table 3.6.3-6.

Table 3.6.3-6 – CursorOver Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX,PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX,SizeY	ulong x 2	32 x 2	A661_SIZE_XY
StyleSet	ushort	16	A661_STYLE_SET

3.6.4 Focus Navigation Widgets

It is often required to move the cursor focus among widgets residing in different layers, including layers owned by different User Applications. Two ways of supporting this kind of Focus navigation are presented here. One method makes use of a single widget (FocusLink). The other method uses two widgets (FocusIn,

FocusOut). For both methods the OEM and CDS provider must take care to manage the configuration of the widgets in the two layers.

Note: The choice of which solution to use is implementation dependent. It is not necessary to implement both solutions.

3.6.4.1 FocusLink

Categories:

- Utility

Description:
The FocusLink widget is used to define a NextFocusedWidget sequence that crosses between two separate layers. The interactive widgets and the FocusLinks are chained together using the NextFocusedWidget parameters in the normal way.

Figure 3.6.4.1 shows an example of two layers with FocusLinks used to join the focus sequences. To the Pilot the focus would move through the sequence 17, 18, 19, 58, 59. Although the FocusLinks (20 and 57) exist in the chain of NextFocusedWidgets, the highlight box does not “focus” on these items.

The NextLayerConnectorRef parameter is used to point to the second layer in the same way as a ConnectorReference is used for the Connector widget.

The value of the NextFocusedWidget is used to point to the next widget in the focus sequence. Note that for the FocusLink the next widget may be on another UA’s layer.

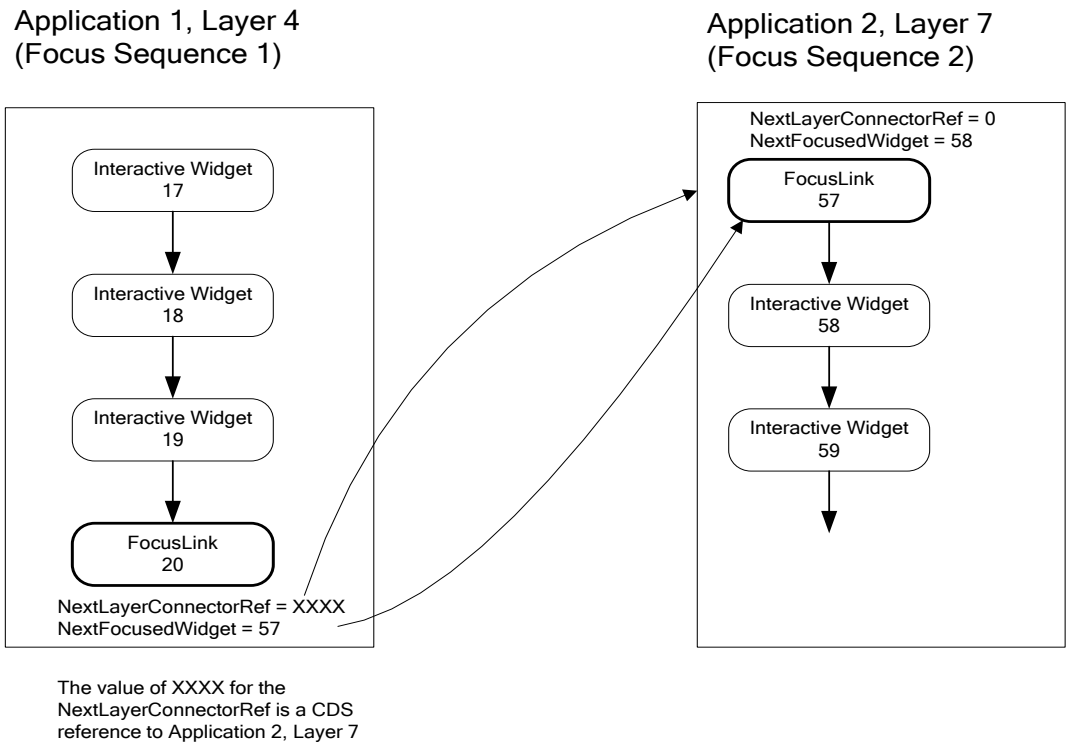


Figure 3.6.4.1 – FocusLink

Restriction:
None

FocusLink Parameters are defined in Table 3.6.4.1-1.

Table 3.6.4.1-1 – FocusLink Parameters

Parameters	Change	Description
WidgetType	D	A661_FOCUS_LINK
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextLayerConnectorRef	D	CDS reference to the linked layer and/or FocusLink widget associated with this focus sequence. The resolution of the link is a CDS configuration issue.
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.

FocusLink Creation Structure is defined in Table 3.6.4.1-2.

Table 3.6.4.1-2 – FocusLink Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_LINK
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	16	0
UnusedPad	N/A	32	0
NextLayerConnectorRef	ushort	16	
NextFocusedWidget	ushort	16	

Table 3.6.4.1-3 – FocusLink Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET

3.6.4.2 FocusIn

Categories:

- Utility

Description:
The FocusIn widget is used together with the FocusOut widget to define a NextFocusedWidget sequence that crosses between two separate layers. The interactive widgets and the FocusOut/FocusIn widgets are chained together using the NextFocusedWidget parameters in the normal way.

Figure 3.6.4.2 shows an example of two layers with FocusOut/FocusIn used to join the focus sequences. To the Pilot, the tabber would move through the sequence 17, 18, 19, 58, 59. Although the FocusOut (20) and FocusIn (57) exist in the chain of NextFocusedWidgets, the highlight box does not “focus” on these items (since they do not have any graphical representation).

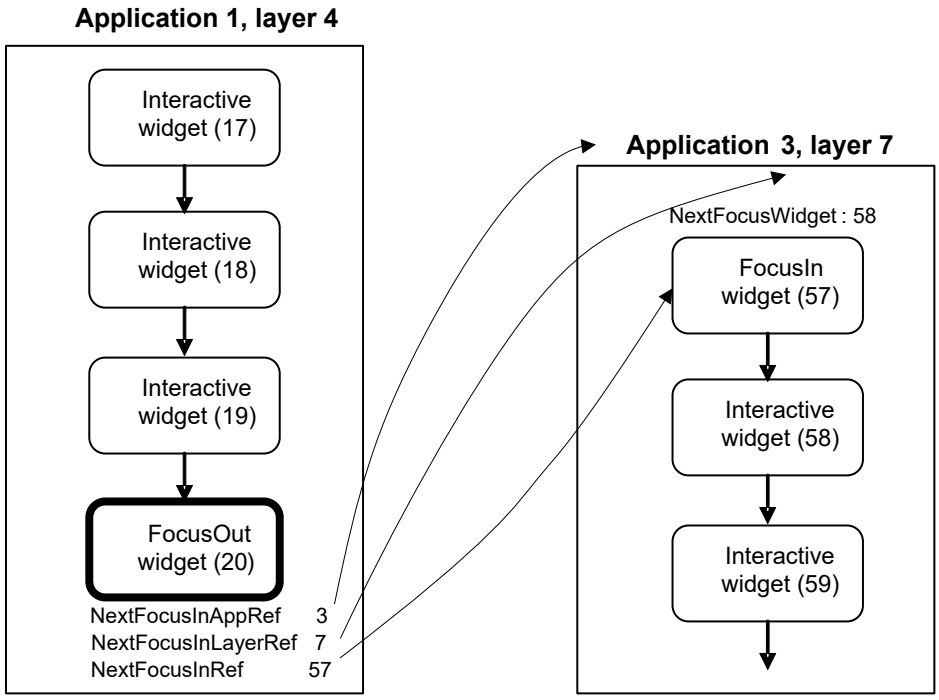


Figure 3.6.4.2 – FocusIn

The connection between the two layers is performed via FocusInRef, an integration level parameter managed by the CDS. The FocusOut widget's NextFocusInRef parameter contains the FocusInRef value identifying the FocusIn widget that will receive the focus. This value is either an identifier or a unique CDS-wide reference.

Each FocusIn widget has a FocusInRef identifier or reference value assigned to it. The CDS supplier is responsible for managing the FocusInRef values and for ensuring that all FocusInRef values are unique. The behavior of the CDS during error conditions is implementation dependent (e.g., if it is found that two FocusIn widgets have been assigned the same FocusInRef value).

Restriction:
None

FocusIn Parameters are defined in Table 3.6.4.2-1.

Table 3.6.4.2-1 – FocusIn Parameters

Parameters	Change	Description
WidgetType	D	A661_FOCUS_IN
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.

FocusIn Creation Structure is defined in Table 3.6.4.2-2.

Table 3.6.4.2-2 – FocusIn Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_IN
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
NextFocusedWidget	ushort	16	

Table 3.6.4.2-3 – FocusIn Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET

3.6.4.3 FocusOut

Categories:

- Utility

Description:
The FocusOut widget is used together with the FocusIn widget to define a NextFocusedWidget sequence that crosses between two separate layers. The interactive widgets and the FocusOut/FocusIn widgets are chained together using the NextFocusedWidget parameters in the normal way.

Figure 3.6.4.3 shows an example of two layers with FocusOut/FocusIn used to join the focus sequences. To the Pilot, the tabber would move through the sequence 17, 18, 19, 58, 59. Although the FocusOut (20) and FocusIn (57) exist in the chain of NextFocusedWidgets, the highlight box does not “focus” on these items (since they do not have any graphical representation).

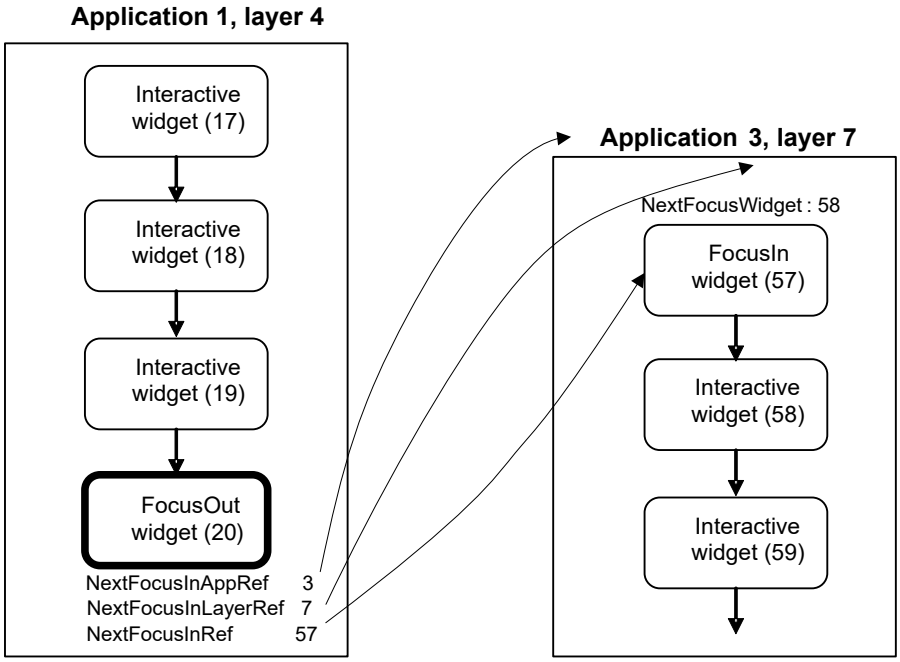


Figure 3.6.4.3 – FocusOut

The connection between the two layers is performed via NextFocusInAppId, NextFocusInLayerId, and NextFocusInId.

Restriction:
None

FocusOut Parameters are defined in Table 3.6.4.3-1.

Table 3.6.4.3-1 – FocusOut Parameters

Parameters	Change	Description
WidgetType	D	A661_FOCUS_OUT
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
NextFocusInAppId	DR	Identifier of the other Application (ApplIdent defined in UADF).
NextFocusInLayerId	DR	Identifier of the other Application layer.
NextFocusInId	DR	Identifier of the FocusIn widget that will receive the focus when the focus reaches the FocusOut widget

FocusOut Creation Structure is defined in Table 3.6.4.3-2.

Table 3.6.4.3-2 – FocusOut Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_FOCUS_OUT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
NextFocusInAppId	ushort	16	
NextFocusInLayerId	uchar	8	
UnusedPad	N/A	8	0
NextFocusInId	ushort	16	

Available SetParameter identifiers and associated data structure are defined in Table 3.6.4.3-3.

Table 3.6.4.3-3 – FocusOut Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NextFocusInAppId	ushort	16	A661_NEXT_FOCUS_IN_APP_ID
NextFocusInLayerId	uchar	8	A661_NEXT_FOCUS_IN_LAYER_ID
NextFocusInId	ushort	16	A661_NEXT_FOCUS_IN_ID

3.6.5 SizeToFitContainer

Categories:

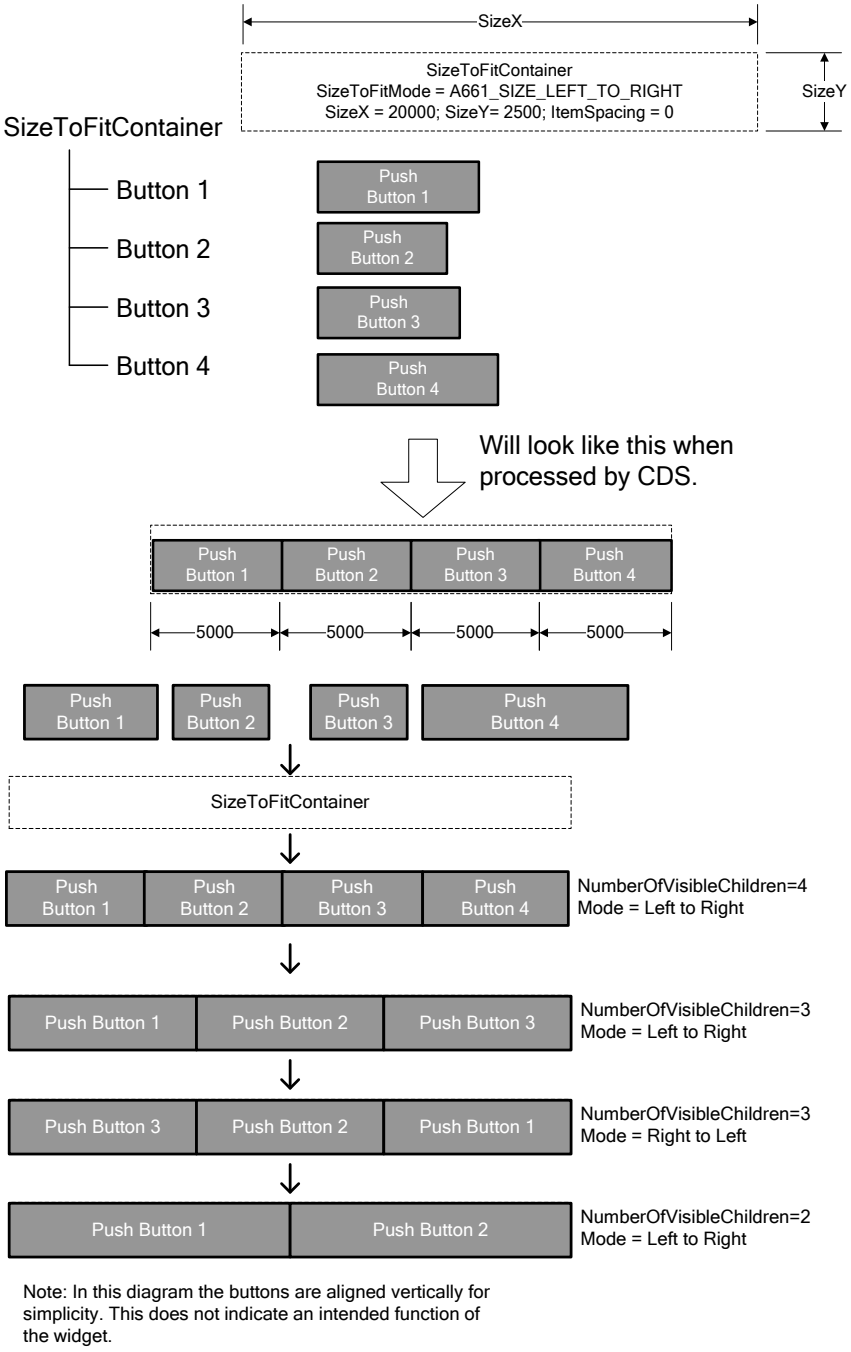
- Container
- Graphical Representation

Description:
This widget is used to stretch/shrink its children such that they all have the same SizeX or SizeY and are arranged either horizontally or vertically within the bounds of the SizeToFitContainer widget. The child widgets can also be configured so that they are ordered from top to bottom, left to right, etc.

This is typically useful when constructing menus or other list of controls, while still allowing the UA developer to control the nature of the interactive widgets that are to

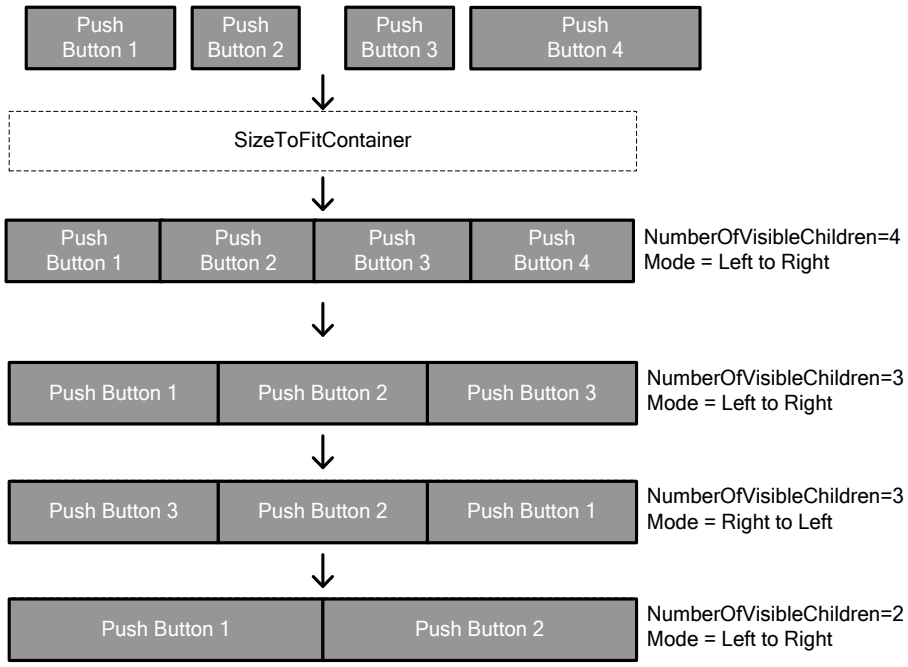
be sized. The widget function can be equally applied to non-interactive widgets such as GpRectangles, etc.

Its purpose is to layout child widgets either in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the SizeToFitContainer.



Note: In this diagram the buttons are aligned vertically for simplicity. This does not indicate an intended function of the widget.

Figure 3.6.5-1 – SizeToFit Example



Note: In this diagram the buttons are aligned vertically for simplicity. This does not indicate an intended function of the widget.

Figure 3.6.5-2 – SizeToFit Example

When the size to fit mode is operating horizontally, the widget SizeX and PosX values are computed.

When the size to fit mode is operating vertically, the widget SizeY and PosY values are computed.

If a horizontal (Size X) sizing mode is selected, the effect on SizeY of the child widgets is implementation dependent. Similarly, if a vertical (SizeY) sizing mode is selected, the effect on SizeX of the child widgets is implementation dependent.

The function of the SizeToFitContainer only applies to the first “n” children whose visibility is set to true, where “n” is equal to NumberOfVisibleChildren. If a child widget’s visibility is set to false it is not considered when counting the NumberOfVisibleChildren.

The SizeToFitContainer only affects its immediate children. It does this even if the Size and/or Pos attributes of the children are definition time only.

The order of widgets is defined by their order in the Definition File. The order direction is determined by the SizeToFitMode parameter.

The function of the widget can be suspended, and the last computed sizes retained by changing the size to fit mode from any fit mode to “no sizing”. When the function is suspended in this way, the child widgets retain their last computed size.

Although the SizeToFitContainer is listed as having a graphical representation, typically it will not appear on the display. Its purpose is to lay out child widgets either

in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the SizeToFitContainer.

An optional item spacing can be applied between the child widgets. The optional item spacing is only applied in the direction indicated by the sizing mode.

In the example shown in Figure 3.6.5-3, the first SizeToFitContainer 1 has two visible children, one of which is another SizeToFitContainer 2. SizeToFitContainer 1 allocated half of the horizontal space to Button 1 and half to SizeToFitContainer 2. SizeToFitContainer 2 then sub-allocates is space to its own three children. Each one being sized to be 33.33% of the space allocated to SizeToFitContainer 2.

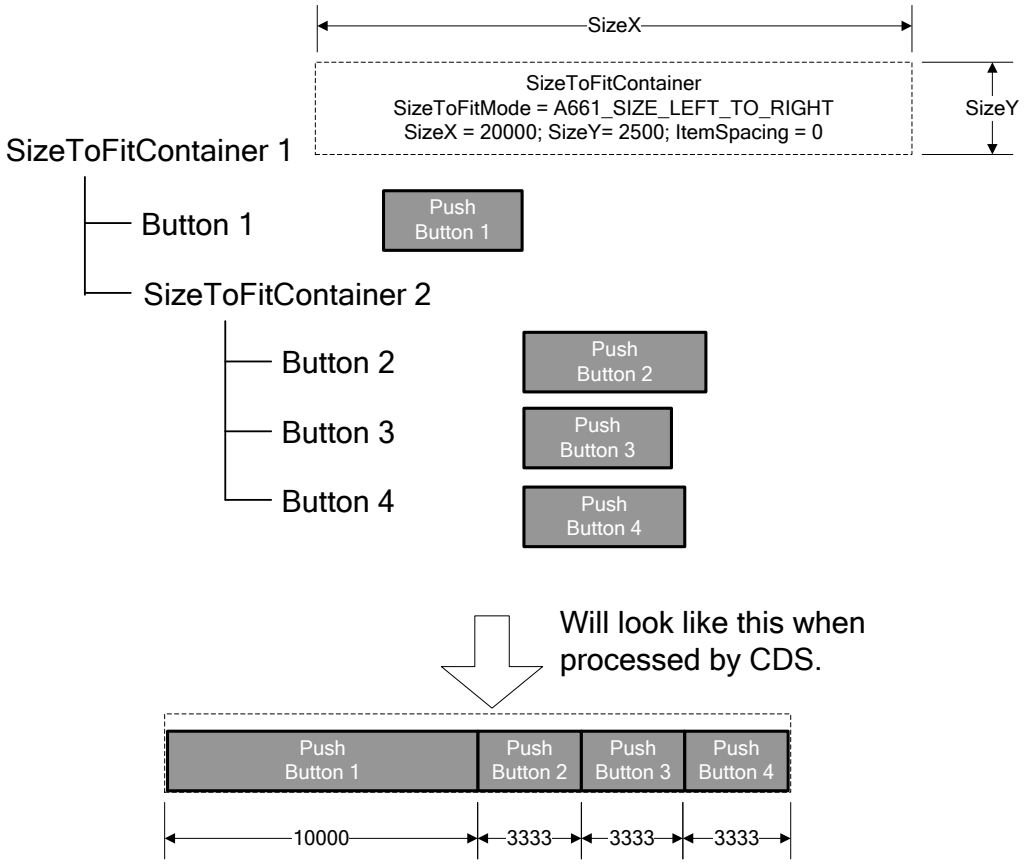


Figure 3.6.5-3 – SizeToFit Example

Nesting of SizeToFitContainers is allowed, but the effect is implementation dependent.

Restrictions:
Any widget that has a SizeX / Size Y can be a child of this widget, although its intended use is that is should only contain basic widgets such as CheckButton, Label, LabelComplex, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ToggleButton, SymbolPushButton, SymbolToggleButton, and SelectionListButton. If complex widgets are made children of this widget, overly complicated nesting should be avoided.

SizeToFitContainer Parameters are defined in Table 3.6.5-1.

Table 3.6.5-1 – SizeToFitContainer Parameters

Parameters	Change	Description
WidgetType	D	A661_SIZE_TO_FIT_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget's descendants to be interactive
Visible	DR	Visibility of the widget
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The width of the widget.
SizeY	DR	The height of the widget.
NumberOfVisibleChildren	DR	Defines the first "n" visible children that to which the sizing function is applied.
SizeToFitMode	DR	Determines how the children in the container are sized to fit. A661_SIZE_TOP_DOWN A661_SIZE_BOTTOM_UP A661_SIZE_LEFT_TO_RIGHT A661_SIZE_RIGHT_TO_LEFT A661_NO_SIZING
ItemSpacing	DR	Defines the spacing between the widgets in 1/100th mm, applied in the axis defined by SizeToFitMode, or the last axis if "no sizing" mode is selected.

SizeToFitContainer Creation Structure is defined in Table 3.6.5-2.

Table 3.6.5-2 – SizeToFitContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SIZE_TO_FIT_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
NumberOfVisibleChildren	ushort	16	
SizeToFitMode	uchar	8	
UnusedPad	N/A	8	0
ItemSpacing	ulong	32	

Available SetParameter identifiers and associated data structure are defined in Table 3.6.5-3.

Table 3.6.5-3 – SizeToFitContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfVisibleChildren	ushort	16	A661_NUMBER_OF_VISIBLE_CHILDREN
SizeToFitMode	uchar	8	A661_SIZE_TO_FIT_MODE
ItemSpacing	ulong	32	A661_ITEM_SPACING

3.6.6 ShuffleToFitContainer

Categories:

- Container
- Graphical Representation

Description:
The ShuffleToFitContainer is used to arrange all its child widgets in a continuous vertical or horizontal stack. Typically, this is used in menus where an item may only be visible some of the time. The remaining items below it are then shuffled up to fill the gap left by the widget(s) that is invisible.

The purpose of the widget is to layout its child widgets either in a horizontal row or a vertical column. The contained widgets are positioned with respect to the PosX, PosY of the ShuffleToFitContainer.

When the widgets are shuffled horizontally, their PosX values are computed. When the widgets are shuffled vertically, their PosY values are computed.

If horizontal shuffling is selected, the effect on PosY of the child widgets is implementation dependent. Similarly, if vertical shuffling is selected, the effect on PosX of the child widgets is implementation dependent.

The function of the ShuffleToFitContainer only applies to the first “n” children whose visibility is set to true, where “n” is equal to NumberOfVisibleChildren. If a child widget’s visibility is set to false, it is not considered when counting the NumberOfVisibleChildren.

The function of the widget can be suspended, and the last computed positions retained by changing the shuffle to fit mode from any fit mode to “no shuffle.” When the function is suspended in this way, the child widgets retain their last computed PosX/Y.

Optional item spacing can be applied between the child widgets. The optional item spacing is only applied in the direction indicated by the fit mode.

The ShuffleToFitContainer only affects the PosX/Y of widgets; it does not affect their Size.

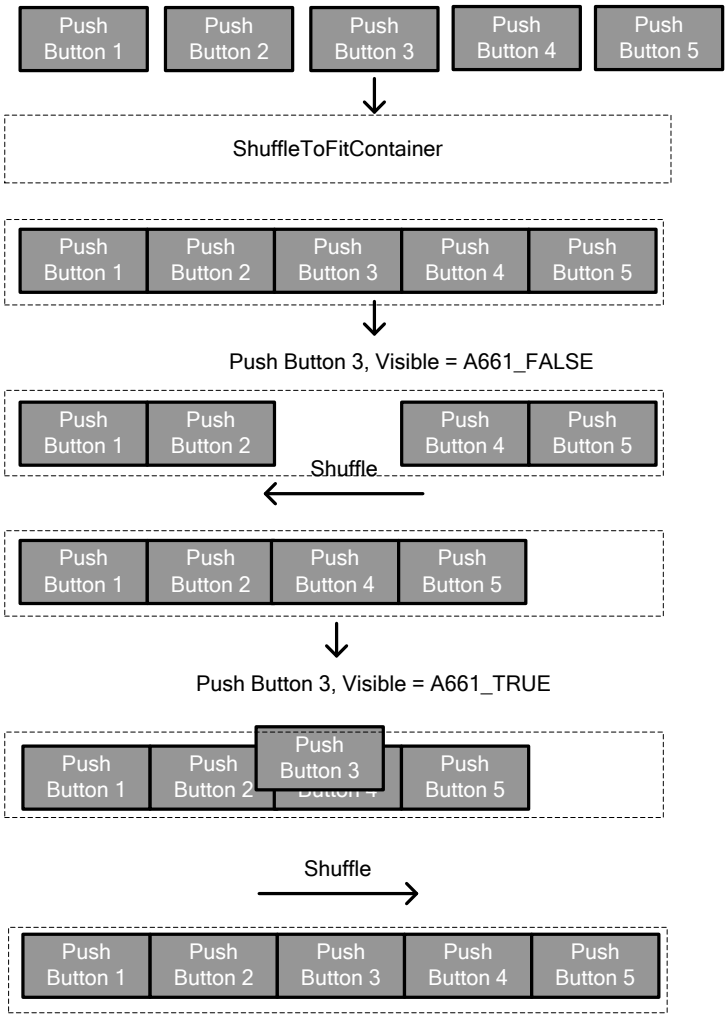


Figure 3.6.6-1 – ShuffleToFit Example

Nesting of ShuffleToFitContainers is allowed, but the effect is implementation dependent.

Restriction:
Any widget that has a SizeX / Size Y can be a child of this widget, although its intended use is that is should only contain basic widgets such as CheckButton, Label, LabelComplex, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ToggleButton, SymbolPushButton, SymbolToggleButton, and SelectionListButton. If complex widgets are made children of this widget, overly complicated nesting should be avoided.

ShuffleToFitContainer Parameters are defined in Table 3.6.6-1.

Table 3.6.6-1 – ShuffleToFitContainer Parameters

Parameters	Change	Description
WidgetType	D	A661_SHUFFLE_TO_FIT_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget's descendants to be interactive
Visible	DR	Visibility of the widget
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The width of the widget.
SizeY	DR	The height of the widget.
NumberOfVisibleChildren	DR	Defines the first "n" visible children that to which the sizing shuffle function is applied.
ShuffleToFitMode	DR	Determines how the children in the container are re-positioned. A661_SHUFFLE_UP A661_SHUFFLE_DOWN A661_SHUFFLE_LEFT A661_SHUFFLE_RIGHT A661_NO_SHUFFLE
ItemSpacing	DR	Defines the spacing between the widgets in 1/100th mm, applied in the axis defined by ShuffleToFitMode.
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.

ShuffleToFitContainer Creation Structure is defined in Table 3.6.6-2.

Table 3.6.6-2 – ShuffleToFitContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SHUFFLE_TO_FIT_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
NumberOfVisibleChildren	ushort	16	
ShuffleToFitMode	uchar	8	A661_SHUFFLE_UP A661_SHUFFLE_DOWN A661_SHUFFLE_LEFT A661_SHUFFLE_RIGHT A661_NO_SHUFFLE
UnusedPad	N/A	8	0
ItemSpacing	long	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.6.6-3.

Table 3.6.6-3 – ShuffleToFitContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfVisibleChildren	ushort	16	A661_NUMBER_OF_VISIBLE_CHILDREN
ShuffleToFitMode	uchar	8	A661_SHUFFLE_TO_FIT_MODE
StyleSet	ushort	16	A661_STYLE_SET
ItemSpacing	ulong	32	A661_ITEM_SPACING

3.7 Widgets Added for Supplement 4

3.7.1 SymbolPushButton

Categories:

- Interactive
- Graphical representation
- Text string

Description:
A SymbolPushButton widget is a PushButton including a symbol and possibly a text string.

Note: When aligning the symbol to the button widget, the symbol representation's bounding rectangle should be used; see the BOUNDING_RECT command in Section 5.2.4.14.

Restriction:
None

SymbolPushButton Parameters are defined in Table 3.7.1-1.

Table 3.7.1-1 – SymbolPushButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SYMBOL_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 interactive
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	DR	The X dimension size (width) of the widget

Parameters	Change	Description
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	Alignment of the text within the label area of the widget: LEFT, RIGHT, CENTER
LabelString	DR	Label of the SymbolPushButton
SymbolReference	DR	Reference of the symbol
SymbolPosition	DR	The string position depends on the symbol position: CENTER, LEFT, RIGHT, TOP, BOTTOM
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

SymbolPushButton Creation Structure is defined in Table 3.7.1-2.

Table 3.7.1-2 – SymbolPushButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SYMBOL_PUSH_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
SymbolReference	ushort	16	
MaxStringLength	ushort	16	
SymbolPosition	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.

SymbolPushButton Event Structures: A661_EVT_SELECTION are defined in Table 3.7.1-3.

Table 3.7.1-3 – SymbolPushButton 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 3.7.1-4.

Table 3.7.1-4 – SymbolPushButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
LabelString	string	{32}+	A661_STRING
SymbolReference	ushort	16	A661_SYMBOL_REFERENCE
StyleSet	ushort	16	A661_STYLE_SET
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
SymbolPosition	**uchar**	**8**	**A661_SYMBOL_POSITION**

3.7.2 SymbolToggleButton

Categories:

- Graphical representation
- Interactive
- Text string

Description:
A SymbolToggleButton widget is a button with two stable states with a symbol and possibly text.

Note: When aligning the symbol to the button widget, the symbol representation's bounding rectangle should be used; see the BOUNDING_RECT command in Section 5.2.4.14.

Restriction:
None

SymbolToggleButton Parameters are defined in Table 3.7.2-1.

Table 3.7.2-1 – SymbolToggleButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SYMBOL_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 interactive
ToggleState	DR	Inner state of the widget: SELECTED UNSELECTED
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
AlternateFlag	DR	True: Use of the two strings (and two symbols) according to the inner state. CDS will change the string if the inner state changes. False: “AlternateString” (and AlternateSymbol) is not used. Only parameter “string” (and Symbol) is used for the two inner states.
LabelString	DR	Label of the widget Label used for UNSELECTED state
AlternateLabelString	DR	Label of the widget Label used for SELECTED state
SymbolReference	DR	Symbol on the widget Symbol used for UNSELECTED state
AlternateSymbolReference	DR	Symbol on the widget Symbol used for SELECTED state
Alignment	DR	Alignment of the text within the label area of the widget: LEFT, RIGHT, CENTER
SymbolPosition	DR	The string position depends on the Symbol position: CENTER, LEFT, RIGHT, TOP, BOTTOM
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

SymbolToggleButton Creation Structure is defined in Table 3.7.2-2.

Table 3.7.2-2 – SymbolToggleButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SYMBOL_TOGGLE_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
AlternateFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
SymbolReference	ushort	16	
AlternateSymbolReference	ushort	16	
SymbolPosition	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.

SymbolToggleButton Event Structures: A661_EVT_STATE_CHANGE are defined in Table 3.7.2-3.

Table 3.7.2-3 – SymbolToggleButton Event Structures: A661_EVT_STATE_CHANGE

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

Available SetParameter identifiers and associated data structure are defined in Table 3.7.2-4.

Table 3.7.2-4 – SymbolToggleButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
ToggleState	uchar	8	A661_INNER_STATE_TOGGLE
StyleSet	ushort	16	A661_STYLE_SET
SymbolReference	ushort	16	A661_SYMBOL_REFERENCE
AlternateSymbolReference	ushort	16	A661_ALTERN_SYMBOL_REFERENCE
LabelString	string	{32}+	A661_STRING
AlternateLabelString	string	{32}+	A661_STRING_ALTERNATE
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
SymbolPosition	**uchar**	**8**	**A661_SYMBOL_POSITION**
AlternateFlag	**uchar**	**8**	**A661_ALTERNATE_FLAG**

3.7.3 PopUpPanelButton

Categories:

- Container
- Graphical Representation
- Interactive
- Text String

Description:
The PopUpPanelButton has a static part (the Button), which, when selected by a crew member, causes the display of the popup part of the PopUpPanelButton.

The popup part of the PopUpPanelButton is similar to a PopUpPanel.

The UA or CDS can define the position of the popup part according to the PositionFlag value.

The children of this widget are positioned relative to the popup part of the PopUpPanelButton widget (relative to the PopupPosX and PopupPosY parameters).

The method for closing the popup part of the widget is implementation dependent.

The popup part of the PopUpPanelButton widget has clipping capability (using the PopupSizeX and PopupSizeY parameters).

Restriction:
None

COMMENTARY

The concept of automatic motion of the focus does not apply because there are no selection or confirmation events for this widget.

PopUpPanelButton Parameters are defined in Table 3.7.3-1.

Table 3.7.3-1 – PopUpPanelButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_POP_UP_PANEL_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 interactive
StyleSet	DR	Reference to predefined graphical characteristics inside CDS
PosX	DR	The X position of the widget (the button) reference point
PosY	DR	The Y position of the widget (the button) reference point
SizeX	DR	The X dimension size (width) of the widget (the button)
SizeY	DR	The Y dimension size (height) of the widget (the button)
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
Specific parameters for the button		
MaxStringLength	D	**Maximum size in bytes of the label text including the NULL terminator.**
Alignment	DR	Alignment of the text within the button's label area of the widget LEFT RIGHT CENTER
LabelString	DR	Label of the button
Picture Reference	DR	Reference of the picture to be displayed on the button
PicturePosition	DR	The string position depends on the picture position: CENTER LEFT RIGHT TOP BOTTOM
Specific Parameters of PopUp		
PopupPosX	DR	The X position of the popup part reference point
PopupPosY	DR	The Y position of the popup part reference point
PopupSizeX	DR	The X dimension size (width) of the popup part
PopupSizeY	DR	The Y dimension size (height) of the popup part
UAPositionFlag	DR	TRUE: UA defined position FALSE: Position defined by CDS using MouseClick location
AutomaticClosure	DR	TRUE: with the automatic closure upon a click outside the PopUpPanel FALSE: without the automatic closure upon a click outside the PopUpPanel

PopUpPanelButton Creation Structure is defined in Table 3.7.3-2.

Table 3.7.3-2 – PopUpPanelButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_POP_UP_PANEL_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
PopupPosX	long	32	
PopupPosY	long	32	
PopupSizeX	ulong	32	
PopupSizeY	ulong	32	
MaxStringLength	ushort	16	
PictureReference	ushort	16	
PicturePosition	uchar	8	A661_CENTER A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
Alignment	uchar	8	A661_LEFT A661_CENTER A661_RIGHT
UAPositionFlag	uchar	8	A661_FALSE A661_TRUE
AutomaticClosure	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

The specific event sent by the PopUpPanelButton to the owner application is defined in Table 3.7.3-3.

Table 3.7.3-3 – PopUpPanelButton Event Structures: A661_EVT_POPUP_PANEL_CLOSED

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

Available SetParameter identifiers and associated data structure are defined in Table 3.7.3-4.

Table 3.7.3-4 – PopUpPanelButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
PosX	**long**	**32**	**A661_POS_X**
PosY	**long**	**32**	**A661_POS_Y**
PosX, PosY	**long x 2**	**32 x 2**	**A661_POS_XY**
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
LabelString	string	{32}+	A661_STRING
PictureReference	ushort	16	A661_PICTURE_REFERENCE
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**
PopupPosX	**long**	**32**	**A661_POPUP_POS_X**
PopupPosY	**long**	**32**	**A661_POPUP_POS_Y**
PopupSizeX	**ulong**	**32**	**A661_POPUP_SIZE_X**
PopupSizeY	**ulong**	**32**	**A661_POPUP_SIZE_Y**
UAPositionFlag	**uchar**	**8**	**A661_UA_POSITION_FLAG**
AutomaticClosure	**uchar**	**8**	**A661_AUTO_CLOSURE**

3.8 Widgets Added for Supplement 5

3.8.1 GpPolyline

Categories:

- Graphical Representation

Description:
The graphical primitive GpPolyline widget enables the definition of a sequence of connected lines defined by a series of vertices.

Restriction:
None

GpPolyline Parameters are defined in Table 3.8.1-1.

Table 3.8.1-1 – GpPolyline Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GP_POLYLINE
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		

Parameters	Change	Description
ColorIndex	DR	Color index of the line. See Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Closed	DR	A661_TRUE: an extra line is drawn from the last vertex to the first vertex. A661_FALSE: no extra line is drawn from the last vertex to the first vertex.
MaxNumberOfVertices	D	The maximum number of vertices to be defined for the widget.
NumberOfVertices	DR	The number of vertices currently defined for the widget.
Vertices	DR	The array of vertices as (X,Y) pairs.

GpPolyline Creation Structure is defined in Table 3.8.1-2.

Table 3.8.1-2 – GpPolyline Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_POLYLINE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
ColorIndex	uchar	8	(valid palette index)
Halo	uchar	8	A661_FALSE A661_TRUE A661_SAME_LEVEL
Closed	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0
MaxNumberOfVertices	ushort	16	Must be greater than or equal to 2.
NumberOfVertices	ushort	16	Must be greater than or equal to 2, and must be less than or equal to MaxNumberOfVertices. If MaxNumberOfVertices is 2 and Closed is A661_TRUE, the behavior is implementation dependent.
Vertices	array of (long, long)	64*NumberOfVertices	There are NumberOfVertices of (x, y) pairs.

The GpPolyline widget does not send any event.

Available GpPolyline identifiers and associated data structure are defined in Table 3.8.1-3.

Table 3.8.1-3 – GpPolyline Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
ColorIndex	uchar	8	A661_COLOR_INDEX
NumberOfVertices	ushort	16	A661_NUMBER_OF_ENTRIES
Vertices [up to MaxNumberOfVertices]	long x 2	{64}+	A661_VERTICES_ARRAY
Halo	**uchar**	**8**	**A661_HALO**
Closed	**uchar**	**8**	**A661_CLOSED**

3.8.2 PagingContainer

Categories:

- Container
- Graphical Representation
- Interactive

Description:
The PagingContainer provides a means for the CDS to manage the exclusive display of its children one at a time, whereas existing widgets, such as the MutuallyExclusiveContainer, requires this to be managed by the UA. Switching between children is performed through UA command or capturing inputs from user interaction. The PagingContainer provides a mechanism to cycle through a series of graphical representations or textual data at runtime, as a means of driving animations or providing graphical feedback to the user.

A UA can make a child visible in the container by setting the VisibleChild parameter. Only the widget in the container that has the WidgetIdent that matches the VisibleChild will be processed for display. The visibility of the child is controlled by the child's Visible parameter and is not changed by changing the VisibleChild parameter. For a child in the container to be visible the VisibleChild must be set to the child's WidgetIdent and the Visible parameter for the widget must be True.

Normal rules for visibility and enabling of an ancestor widget apply to the PagingContainer.

The contained widgets are positioned with respect to the PosX and PosY of the PagingContainer. Its size is only used for determining if inputs fall within the region of display space to be interactive for the PagingContainer. Both the UA and the user can cause the VisibleChild parameter to change.

User interaction can be achieved through any mechanism that results in the interpretation of an increment or decrement of the child position index, based upon the order of the children in the layer definition. According to this new position, VisibleChild is updated with the corresponding child's WidgetIdent.

Restriction: None

PagingContainer Parameters are defined in Table 3.8.2-1.

Table 3.8.2-1 – PagingContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_PAGING_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Enable	DR	Ability of the widget to be interactive.
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		
WrappingType	DR	Defines what type of wrapping is allowed (i.e., max to min, min to max, both, none)
PagingControlPosition	DR	Defines the location of the PagingControls
FinePageDelta	DR	Fine increment step for modification of the visible child.
CoarsePageDelta	DR	Coarse increment step for modification of the visible child.
VisibleChild	DR	Identifier of the widget to be made visible. A value of 0 means that no widget is visible.
ReportVisibleChild	D	If True, CDS will report changes of the visible child ident (A661_EVT_VISIBLE_CHILD).
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

PagingContainer Creation Structure is defined in Table 3.8.2-2.

Table 3.8.2-2 – PagingContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_PAGING_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
WrappingType	uchar	8	A661_WRAP_NONE A661_WRAP_BOTH A661_WRAP_FIRST TO LAST A661_WRAP_LAST_TO_FIRST
PagingControlPosition	uchar	8	A661_ABSENT A661_BOTTOM_CENTER A661_BOTTOM_LEFT A661_BOTTOM_RIGHT

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
			A661_CENTER A661_LEFT A661_RIGHT A661_TOP_CENTER A661_TOP_LEFT A661_TOP_RIGHT
FinePageDelta	ushort	16	
CoarsePageDelta	ushort	16	
VisibleChild	ushort	16	
ReportVisibleChild	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE

The specific event sent by the PagingContainer to the owner application is defined in Table 3.8.2-3.

Table 3.8.2-3 – PagingContainer Events Structure: A661_EVT_VISIBLE_CHILD

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_VISIBLE_CHILD
VisibleChild	ushort	16	Widget Ident of VisibleChild

Available PagingContainer identifiers and associated data structure are defined in Table 3.8.2-4.

Table 3.8.2-4 – PagingContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
VisibleChild	ushort	16	A661_VISIBLE_CHILD
EntryValidation	uchar	8	A661_ENTRY_VALID
WrappingType	**uchar**	**8**	**A661_PAGE_WRAP_TYPE**
PagingControlPosition	**uchar**	**8**	**A661_PAGE_CONTROL_POSITION**
FinePageDelta	**ushort**	**16**	**A661_FINE_PAGE_DELTA**
CoarsePageDelta	**ushort**	**16**	**A661_COARSE_PAGE_DELTA**

3.8.3 NumericReadout

Categories:

- Graphical representation
- Text string

Description:
A NumericReadout widget is non-interactive and consists of a numeric value and a

text legend. The formatting of the numeric value is performed by the CDS and not the UA.

Restriction:
None

NumericReadout Parameters are defined in Table 3.8.3-1.

Table 3.8.3-1 – NumericReadout Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_NUMERIC_READOUT
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
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
Value	DR	Value displayed by the widget
MotionAllowed	D	Capability to change PosX, PosY, RotationAngle at runtime
RotationAngle	DR	Angle at which NumericReadout is displayed relative to its origin Refer to Angles defined in Section 2.3.4.2
FormatString	DR	String describing the format of the numeric field. The string is composed of any authorized characters. Some of them will be interpreted to format the value. **See Section 3.2.6 "Formatted Numeric Values" for the formatting.**
MaxFormatStringLength	D	**Maximum size in bytes (including the NULL terminator) of FormatString.**
Alignment	DR	Justification of the numeric value within the widget display area: BottomCenter BottomLeft BottomRight Center Left Right TopCenter TopLeft TopRight
MaxLegendStringLength	D	Max string size for the legend (MaxLegendStringLength > 0)
LegendAreaSizeX	DR	The X dimension (size) of the legend
LegendString	DR	Legend associated to the numeric value
LegendPosition	DR	Position of the legend in comparison with the numeric value: Left Right Top Bottom

NumericReadout Creation Structure is defined in Table 3.8.3-2.

Table 3.8.3-2 – NumericReadout Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_NUMERIC_READOUT
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
MotionAllowed	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	
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	8	0
Value	float	32	
LegendAreaSizeX	ulong	32	
MaxFormatStringLength	uchar	8	
MaxLegendStringLength	uchar	8	
LegendPosition	uchar	8	A661_LEFT A661_RIGHT A661_TOP A661_BOTTOM
UnusedPad	NA	8	0
FormatString	string	8 * string length	
LegendString	string	8 * string length + PAD	Followed by zero, one, two, or three extra NULL for alignment on 32 bits.

The NumericReadout widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.3-3.

Table 3.8.3-3 – NumericReadout Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Value	float	32	A661_VALUE
PosX, PosY	long x 2	32 x 2	A661_POS_XY
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
SizeX	**ulong**	**32**	**A661_SIZE_X**
SizeY	**ulong**	**32**	**A661_SIZE_Y**
SizeX, SizeY	**ulong x 2**	**32 x 2**	**A661_SIZE_XY**
StyleSet	ushort	16	A661_STYLE_SET
LegendString	string	{32}+	A661_STRING
FormatString	String	{32}+	A661_FORMAT_STRING
LegendPosition	uchar	8	A661_LEGEND_POSITION
RotationAngle	fr(180)	32	A661_ORIENTATION
Alignment	**uchar**	**8**	**A661_ALIGNMENT**
LegendAreaSizeX	**ulong**	**32**	**A661_LEGEND_AREA_SIZE_X**

3.8.4 MapHorz_Container

Categories:

- Container
- Map management

Description:
The MapHorz_Container is used to position a set of non-interactive widgets on a map using real world coordinates (e.g., Lat/Lng). The X and Y parameters provide this real world position data. Their interpretation is dependent on the MapDataFormat. The screen location resulting from the projection of the X and Y parameters defines the origin of the MapHorz_Container's children. The position and size of the children are still defined in hundredth of millimeters.

The MapHorz_Container rotates its children about the point resulting from the projection of the X and Y parameters. The MapHorz_Container rotates its children according to the Orientation parameter and the Orientation parameter of the ancestor MapHorz. The MapHorz_Container scales its children according to the widgets Range and ScreenRange parameters as well as the Range and ScreenRange parameters of the ancestor MapHorz.

Behavior of this widget before the X, Y, Orientation, Range, and ScreenRange parameters are received is implementation dependent.

The MapHorz_Container sends events to its owner to indicate when its children widgets are onscreen or offscreen. This is calculated based on the Range parameter.

Restriction:
A MapHorz_Container must be in a MapHorz_Source container.

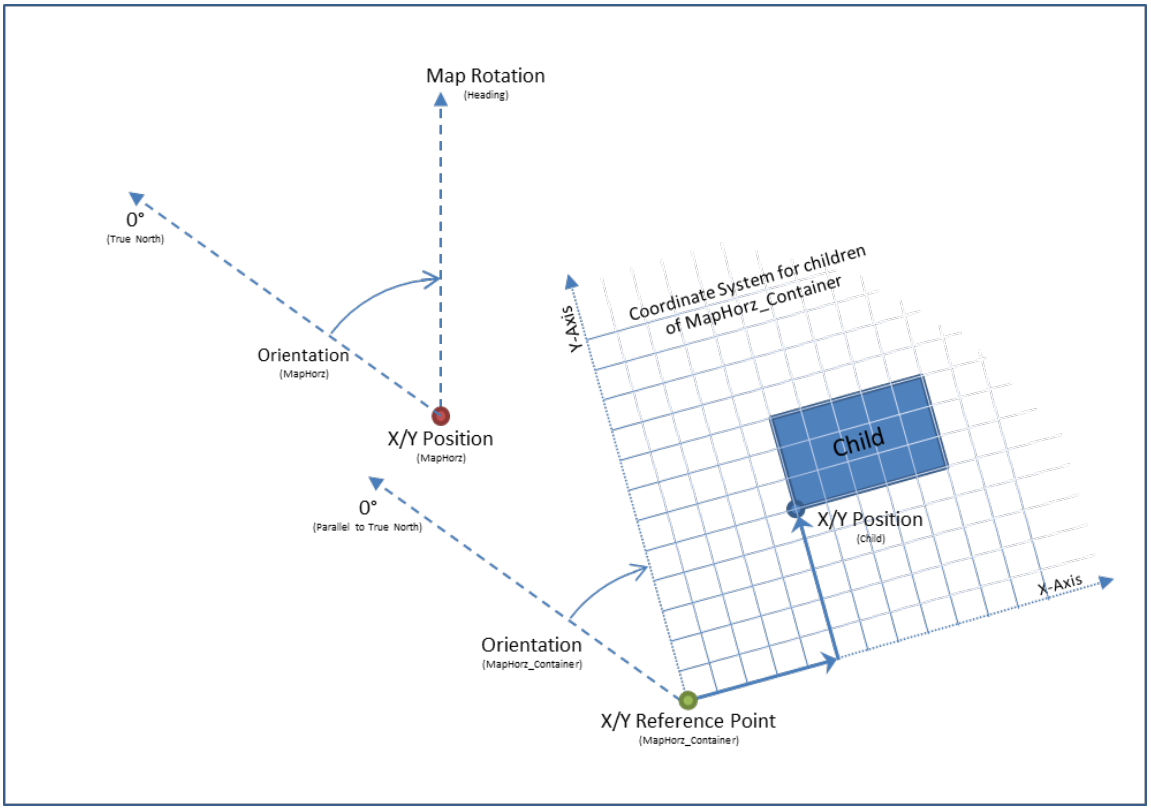


Figure 3.8.4 – MapHorz_Container

MapHorz_Container Parameters are defined in Table 3.8.4-1.

Table 3.8.4-1 – MapHorz_Container Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_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		
X	R	Dependent on MapDataFormat See Table 3.3.24-2b
Y	R	Dependent on MapDataFormat See Table 3.3.24-2b
Orientation	R	Angle between True North and the positive y-axis of the children at the origin of the children (see Reference coordinate system, positive direction: counter-clockwise).
MapSynchronizationNumber	R	See Section 3.2.8.4
Range	R	Geo-referenced range expressed in nm
ScreenRange	R	Distance in screen units (hundredth of mm) equivalent to range

MapHorz_Container Creation Structures are defined in Table 3.8.4-2.

Table 3.8.4-2 – MapHorz_Container Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	8	0
Visible	uchar	8	A661_FALSE A661_TRUE

MapHorz_Container Event Structures: A661_EVT_ONSCREEN is defined in Table 3.8.4-3

Table 3.8.4-3 – MapHorz_Container Event Structures: A661_EVT_ONSCREEN

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ONSCREEN
UnusedPad	N/A	16	0

MapHorz_Container Event Structures: A661_EVT_OFFSCREEN is defined in Table 3.8.4-4

Table 3.8.4-4 – MapHorz_Container Event Structures: A661_EVT_OFFSCREEN

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_OFFSCREEN
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.8.4-5.

Table 3.8.4-5 – MapHorz_Container Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
X	Dependent on MapDataFormat	32	A661_REF_POS_X
Y	Dependent on MapDataFormat	32	A661_REF_POS_Y
X,Y	Dependent on MapDataFormat	32 x 2	A661_REF_POS_XY
Orientation	fr(180)	32	A661_ORIENTATION
Range	fr(32768)	32	A661_RANGE
ScreenRange	ulong	32	A661_SCREEN_RANGE
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER See Section 3.2.8.4

3.8.5 MapHorz_Panel

Categories:

- Container
- Map management
- Graphical Representation

This container can be embedded into a **MapHorz_Source** widget. The X and Y parameters provide this real world position data. Their interpretation is dependent on the MapDataFormat. The screen location resulting from the projection of the X and Y parameters defines the origin of the MapHorz_Panel's children.

Its aim is to provide geographic-aware interactivity, e.g., it is possible to add a few buttons close to a waypoint or a text entry close to an airport.

The unit for the widgets within the container remains the standard hundredth of mm A661 unit. The size of the widget can be used by the CDS to manage specific placement policies.

Table 3.8.5-1 – MapHorz_Panel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_PANEL
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Enable	DR	Ability of the widget's descendants to be interactive
Visible	DR	Visibility of the widget
SizeX	DR	The X dimension size (width) of the widget in hundredths of mm.
SizeY	DR	The Y dimension size (height) of the widget in hundredths of mm.
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
Specific parameters		
X	DR	Dependent on MapDataFormat See Table 3.3.24-2b
Y	DR	Dependent on MapDataFormat See Table 3.3.24-2b
MapSynchronizationNumber	R	See Section 3.2.8.4

MapHorz_Panel Creation Structure is defined in Table 3.8.5-2.

Table 3.8.5-2 – MapHorz_Panel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
X	Dependent on MapDataFormat	32	See Table 3.3.24-2b
Y	Dependent on MapDataFormat	32	See Table 3.3.24-2b
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0

The MapHorz_Panel widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.5-3.

Table 3.8.5-3 – A661_MapHorz_Panel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
X, Y	long x 2	32 x 2	A661_REF_POS_XY
X	long	32	A661_REF_POS_X
Y	long	32	A661_REF_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4

3.8.6 DataScalingLong

Categories:
None

Description:
The objective of this widget is to provide a means for scaling one long value for one widget parameter.

The widget referenced in this DataScalingLong widget must be defined in the Definition File before the DataScalingLong widget.

The algorithm for this widget is defined below:

- If (EndValue = StartValue) then scale = 0
- Else scale = (Value – StartValue) / (EndValue – StartValue)

ParamValue = StartScaledValue + (EndScaledValue – StartScaledValue) * scale

ParamValue is cast to long and clamped to the long range.

If (Clamping) then:

- If (ParamValue < StartScaledValue) then ParamValue = StartScaledValue
- Else if (ParamValue > EndScaledValue) then ParamValue = EndScaledValue

Restrictions:

- The DataScalingLong can only be the child of a layer
- The DataScalingLong cannot refer to a “Definition Only” parameter
- The referred parameter type must be long

DataScalingLong Parameters are defined in Table 3.8.6-1.

Table 3.8.6-1 – DataScalingLong Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_DATA_SCALING_LONG
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget. The only possible parent of the DataScalingLong is the layer; therefore' ParentIdent Value: 0
Specific parameters		
TargetWidgetID	D	Target Widget ID for which the parameter should be scaled
TargetParameterID	D	Target ParameterID for the widget which should be scaled. Its type should be Long
Clamping	DR	Determine if the parameter is clamped to its [StartScaledValue, EndScaledValue] range
StartValue	DR	Start value of the parameter (corresponding to a scale of 0)
EndValue	DR	End value of the parameter (corresponding to a scale of 1)
StartScaledValue	DR	Start value of the parameter after scaling (corresponding to a scale of 0)
EndScaledValue	DR	End value of the parameter after scaling (corresponding to a scale of 1)
Value	R	Value

DataScalingLong Creation Structure is defined in Table 3.8.6-2.

Table 3.8.6-2 – DataScalingLong Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_DATA_SCALING_LONG
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
TargetWidgetID	ushort	16	
TargetParameterID	ushort	16	
Clamping	uchar	8	A661_NOT_CLAMPED A661_BOTH_CLAMPED
Pad	N/A	8	0
StartValue	float	32	
EndValue	float	32	
StartScaledValue	long	32	
EndScaledValue	long	32	

The DataScalingLong widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.6-3.

Table 3.8.6-3 – DataScalingLong Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Clamping	uchar	8	A661_CLAMPING
Value	float	32	A661_VALUE
StartValue	float	32	A661_START_VALUE
EndValue	float	32	A661_END_VALUE
StartScaledValue	long	32	A661_START_SCALED_VALUE
EndScaledValue	long	32	A661_END_SCALED_VALUE

3.8.7 DataScalingULong

Categories:
None

Description:
The objective of this widget is to provide a means for scaling one ulong value for one widget parameter.

The widget referenced in this DataScalingULong widget must be defined in the Definition File before the DataScalingULong widget.

The algorithm for this widget is defined below:

- If (EndValue = StartValue) then scale = 0
- Else scale = (Value – StartValue) / (EndValue – StartValue)

ParamValue = StartScaledValue + (EndScaledValue – StartScaledValue) * scale

ParamValue is cast to ulong and clamped to the ulong range.

If (Clamping) then:

- If (ParamValue < StartScaledValue) then ParamValue = StartScaledValue
- Else if (ParamValue > EndScaledValue) then ParamValue = EndScaledValue

Restrictions:

- The DataScalingULong can only be the child of a layer
- The DataScalingULong cannot refer to a “Definition Only” parameter
- The referred parameter type must be ulong

DataScalingULong Parameters are defined in Table 3.8.7-1.

Table 3.8.7-1 – DataScalingULong Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_DATA_SCALING_ULONG
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget. The only possible parent of the DataScalingULong is the layer; therefore, ParentIdent Value: 0
Specific parameters		
TargetWidgetID	D	Widget ID for which the parameter should be scaled
TargetParameterID	D	ParameterID for the widget which should be scaled. Its type should be Long
Clamping	DR	Determine if the parameter is clamped to its [StartScaledValue, EndScaledValue] range
StartValue	DR	Start value of the parameter (corresponding to a scale of 0)
EndValue	DR	End value of the parameter (corresponding to a scale of 1)
StartScaledValue	DR	Start value of the parameter after scaling (corresponding to a scale of 0)
EndScaledValue	DR	End value of the parameter after scaling (corresponding to a scale of 1)
Value	R	Value

DataScalingULong Creation Structure is defined in Table 3.8.7-2.

Table 3.8.7-2 – DataScalingULong Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_DATA_SCALING_ULONG
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
TargetWidgetID	ushort	16	
TargetParameterID	ushort	16	
Clamping	uchar	8	A661_NOT_CLAMPED A661_BOTH_CLAMPED
Pad	N/A	8	0
StartValue	float	32	
EndValue	float	32	
StartScaledValue	ulong	32	
EndScaledValue	ulong	32	

The DataScalingULong widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.7-3.

Table 3.8.7-3 – DataScalingULong Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Clamping	uchar	8	A661_CLAMPING
Value	float	32	A661_VALUE
StartValue	float	32	A661_START_VALUE
EndValue	float	32	A661_END_VALUE
StartScaledValue	ulong	32	A661_START_SCALED_VALUE
EndScaledValue	ulong	32	A661_END_SCALED_VALUE

3.8.8 DataScalingFR180

Categories:
None

Description:
The objective of this widget is to provide a means for scaling one fr(180) value for one widget parameter.

The widget referenced in this DataScalingFR180 widget must be defined in the Definition File before the DataScalingFR180 widget.

The algorithm for this widget is defined below:

- If (EndValue = StartValue) then scale = 0
- Else scale = (Value – StartValue)/(EndValue – StartValue)

The resulting angle value is calculated clockwise:

ParamValue = StartScaledValue + (EndScaledValue – StartScaledValue) * scale

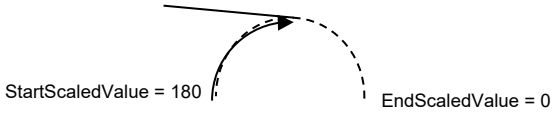


Figure 3.8.8 – Scaled Value

ParamValue is cast to ulong and clamped to the ulong range.

If (Clamping) then:

- If (ParamValue < StartScaledValue) then ParamValue = StartScaledValue
- Else if (ParamValue > EndScaledValue) then ParamValue = EndScaledValue

ParamValue should be wrapped in the [-180, 180] extension by a modulo function

Note: For example, for a ParamValue of 200 after applying the scale, then it will be considered as -160.

Restrictions:

- The DataScalingFR180 can only be the child of a layer
- The DataScaling cannot refer to a “Definition Only” parameter
- The referred parameter type must be fr(180)

DataScalingFR180 Parameters are defined in Table 3.8.8-1.

Table 3.8.8-1 – DataScalingFR180 Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_DATA_SCALING_FR180
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget. The only possible parent of the DataScalingFR180 is the layer; therefore, ParentIdent Value: 0
Specific parameters		
TargetWidgetID	D	Widget ID for which the parameter should be scaled
TargetParameterID	D	ParameterID for the widget which should be scaled. Its type should be Long
Clamping	DR	Determine if the parameter is clamped to its [StartScaledValue, EndScaledValue] range
StartValue	DR	Start value of the parameter (corresponding to a scale of 0)
EndValue	DR	End value of the parameter (corresponding to a scale of 1)
StartScaledValue	DR	Start value of the parameter after scaling (corresponding to a scale of 0)
EndScaledValue	DR	End value of the parameter after scaling (corresponding to a scale of 1)
Value	R	Value

DataScalingFR180 Creation Structure is defined in Table 3.8.8-2.

Table 3.8.8-2 – DataScalingFR180 Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_DATA_SCALING_FR180
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
TargetWidgetID	ushort	16	
TargetParameterID	ushort	16	
Clamping	uchar	8	A661_NOT_CLAMPED A661_BOTH_CLAMPED
Pad	N/A	8	0
StartValue	float	32	
EndValue	float	32	
StartScaledValue	fr(180)	32	
EndScaledValue	fr(180)	32	

The DataScalingFR180 widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.8-3.

Table 3.8.8-3 – DataScalingFR180 Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Clamping	uchar	8	A661_CLAMPING
Value	float	32	A661_VALUE
StartValue	float	32	A661_START_VALUE
EndValue	float	32	A661_END_VALUE
StartScaledValue	fr(180)	32	A661_START_SCALED_VALUE
EndScaledValue	fr(180)	32	A661_END_SCALED_VALUE

3.8.9 BroadcastReceiver

Categories:
None

Description:
The objective of this widget is to provide a means for broadcasting one value to different widgets in one layer. For example, it is possible by using this widget to send only one ColorIndex value and broadcast it to several widgets, which will all change their Color according to this same value.

The data structure to send depends of the type of parameter to broadcast, defined by the ParameterIdent parameter (see Table 4.6-8 – Parameter Types). The list of widgets where to send the value of this parameter is also fixed at definition time through the WidgetStructure parameter. For each widget in the WidgetStructure, the CDS will perform a set on the widget parameter of ID ParameterIdent.

The widgets referenced in this BroadcastReceiver widget must be defined in the Definition File before the BroadcastReceiver widget.

Restrictions:

- The BroadcastReceiver can only be the child of a layer

ParameterIdent must be run-time modifiable for all target widgetsBroadcastReceiver Parameters are defined in Table 3.8.9-1.

Table 3.8.9-1 – BroadcastReceiver Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_BROADCAST_RECEIVER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget. The only possible parent of the BroadcastReceiver is the layer, therefore ParentIdent Value: 0
Specific parameters		
NumberOfFields	D	Number of fields in the structure
ParameterIdent	D	The ParameterIdent used for the BroadcastReceiver.
WidgetStructure	D	List of WidgetIdent for the value to be sent by the UA through the BroadcastReceiver. The number of widgets is defined by the NumberOfFields. The size of this parameter, in bytes, is 2 * NumberOfFields
BroadcastData	R	The data which has to be broadcast

BroadcastReceiver Creation Structure is defined in Table 3.8.9-2.

Table 3.8.9-2 – BroadcastReceiver Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_BROADCAST_RECEIVER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
NumberOfFields	ushort	16	
ParameterIdent	ushort	16	
WidgetStructure[NumberOfFields]	{ushort}+	16*Number of Fields + pad	List: WidgetIdent (16 bits) Padded to be 32 bits aligned.

The BroadcastReceiver widget does not send any events.

Available SetParameter identifiers and associated data structure are defined in Table 3.8.9-3.

Table 3.8.9-3 – BroadcastReceiver 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)
BroadcastData	N/A	{32}+	A661_BROADCAST_DATA	The actual structure of the data depends of the structure of parameter to broadcast, defined in the ParameterIdent parameter, padded to 32 bits

For example, consider a Definition File containing these widgets:

- A661_GP_LINE (WidgetIdent 10)
- A661_GP_LINE (WidgetIdent 15)
- A661_GP_RECTANGLE (WidgetIdent 20)

and one A661_BROADCAST_RECEIVER widget, with the A661_COLOR_INDEX (0xB160) value for the ParameterIdent parameter, and the following WidgetStructure:

- widget id: 10
- widget id: 15
- widget id: 20

The runtime value of the A661_BROADCAST_DATA will be used for these three parameters.

3.8.10 NoServiceMonitor

Categories:

- Container
- Utility

Description:
The NoServiceMonitor widget can be used by the application developer to display information when no communication between the UA and CDS is possible. The NoServiceMonitor is an immediate child of a Layer, and cannot be placed in any other container. Typically, when NoServiceMonitor is used, all widgets in a layer (except utilities like BufferFormat or widgets that are to be displayed regardless) will be descendants of the NoServiceMonitor container, but those descendants associated with the ShowNoServiceIdent parameter are treated differently from descendants not associated with the ShowNoServiceIdent parameter.

The widget referenced by the ShowNoServiceIdent parameter is evaluated when the UA is not capable of driving the layer. The cause(s) and determination of when the NoServiceMonitor is invoked is managed by the CDS, including any timeout duration. Examples include failure of a UA to respond to a layer activation notification, detectable network connection problem between CDS and UA, and lack of a UA to whom CDS can send a layer activation notification.

When the NoServiceMonitor is triggered, only the widget referenced by the ShowNoServiceIdent parameters and its descendants are evaluated for display, and all other descendants of the NoServiceMonitor are not evaluated. The descendant widgets must have their visibility parameters set appropriately in order to be displayed. The layer does not have to be set to visible by the UA for the widget associated with ShowNoServiceIdent and its descendants to be displayed.

When the NoServiceMonitor condition is not invoked, then all other descendants of the NoServiceMonitor are evaluated for display, while the widget referenced by the ShowNoServiceIdent parameter and its descendants are not evaluated for display.

Siblings of a NoServiceMonitor widget are processed independent of the state of the NoServiceMonitor. If the NoServiceMonitor widget is the child of a connected layer, its children must be valid children of the Connector widget’s parent.

COMMENTARY

If Connector widget is included under the ShowNoServiceIdent it is expected that the connected Layer will activate normally when the NoServiceMonitor is triggered.

Restriction:

The widget referenced by ShowNoServiceIdent must be an immediate child of the NoServiceMonitor widget.

NoServiceMonitor Parameters are defined in Table 3.8.10-1.

Table 3.8.10-1 – NoServiceMonitor Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_NO_SERVICE_MONITOR
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Specific parameters		
ShowNoServiceIdent	D	ID of the child widget to be evaluated for display only when the CDS detects loss of UA.

NoServiceMonitor Creation Structure is defined in Table 3.8.10-2.

Table 3.8.10-2 – NoServiceMonitor Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_NO_SERVICE_MONITOR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
ShowNoServiceIdent	ushort	16	
UnusedPad	N/A	32	0

The NoServiceMonitor widget does not send any event, and it does not have any run-time modifiable parameters.

3.9 Widgets Added for Supplement 6

3.9.1 MultiStateButton

Categories:

- Graphical Representation
- Interactive
- Text String

Description:
A MultiStateButton widget is a multiple stable-state button with text. It is similar to the ToggleButton widget but supports more than two states.

Each time the MultiStateButton is selected, the value of ButtonState enumerates through a sequence from 1 to NumberOfStates, incrementing one state upon each selection. If ButtonState is equal to NumberOfStates when selected, the value goes back to 1. A ButtonState of 0 is undefined.

Restriction:

- A MultiStateButton cannot be a child of a RadioBox widget.

MultiStateButton Parameters are defined in Table 3.9.1-1.

Table 3.9.1-1 – MultiStateButton Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MULTI_STATE_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
ButtonState	DR	Inner state of the button, from 1 to NumberOfStates
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused upon crew member validation
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
MaxStringLength	D	**Maximum size in bytes (including the NULL terminator) of each string array entry.**
Alignment	DR	Alignment of the label text within the label area of the widget: LEFT RIGHT CENTER
MaxNumberOfStates	D	Maximum number of selectable states
NumberOfStates	DR	Current number of selectable states (not counting the unselected state). Must be less than or equal to MaxNumberOfStates
LabelStringArray	DR	String array holding the value of the button label text for each state
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing A661_FALSE A661_TRUE

MultiStateButton Creation Structure is defined in Table 3.9.1-2.

Table 3.9.1-2 – MultiStateButton Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MULTI_STATE_BUTTON
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxStringLength	ushort	16	
ButtonState	uchar	8	1 to NumberOfStates
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_LEFT A661_RIGHT A661_CENTER
MaxNumberOfStates	uchar	8	1 to 255
NumberOfStates	uchar	8	1 to MaxNumberOfStates
UnusedPad	N/A	8	0
LabelStringArray[NumberOfStates]	{string}+	{32}+	Each string is ended by a NULL character, which is used as a string separator. The complete string list is followed by zero, one, two or three NULL character(s) to be 32 bits aligned.

The events sent by the MultiStateButton to the owner application are:

Table 3.9.1-3 – MultiStateButton Event Structures: A661_EVT_STATE_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_STATE_CHANGE
ButtonState	uchar	8	1 to NumberOfStates
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.1-4.

Table 3.9.1-4 – MultiStateButton Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
ButtonState	uchar	8	A661_MULTI_STATE_VALUE
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
StyleSet	ushort	16	A661_STYLE_SET
NumberOfStates	uchar	8	A661_NUMBER_OF_STATES
LabelStringArray[up to MaxNumberOfStates]	{string}+	{32}+	A661_STRING_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
AutomaticFocusMotion	uchar	8	A661_AUTO_FOCUS_MOTION
Alignment	uchar	8	A661_ALIGNMENT

3.9.2 KeyboardArea

Categories:

- Interactive

Description:
The KeyboardArea is a transparent rectangular widget. The objective of this widget is to send keyboard key press events from Crew member interactions. The principle is that this KeyboardArea widget defines a zone inside which the key press events are redirected to UA while the Cursor is over the area and when the keyboard inputs are not consumed by another widget.

Note: An input is considered consumed by a widget X when it is made unavailable for any widget to trigger internal behavior after being used by the aforesaid widget X.

KeyboardArea widgets that are drawn on top of other interactive widget do not prevent the widget underneath from generating events. Under these conditions the KeyboardArea widget and the interactive widget underneath will still be able to generate events. This is one of the noted exceptions to the recommendation defined in Section 2.3.5.2 for overlapping widgets.

KeyboardArea Parameters are defined in Table 3.9.2-1.

Table 3.9.2-1 –KeyboardArea Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_KEYBOARD_AREA
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be activated
PosX	DR	The X position of the widget reference point (screen coordinate system)
PosY	DR	The Y position of the widget reference point (screen coordinate system)
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
NumberOfKeys	DR	Number of accessible keys
MaxNumberOfKeys	D	Max number of keys that can be managed by the widget **(must be greater than 0)**
KeyArray	DR	**Array of the key code(s) for which the A661_EVT_KEY event will be transmitted**

KeyboardArea Creation Structure is defined in Table 3.9.2-2.

Table 3.9.2-2 –KeyboardArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range when necessary
WidgetType	ushort	16	A661_KEYBOARD_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
NumberOfKeys	uchar	8	
MaxNumberOfKeys	uchar	8	

CreateParameterBuffer	Type	Size (bits)	Value/Range when necessary
UnusedPad	N/A	24	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
KeyArray [NumberOfKeys]	{ushort}+	16 * NumberOfKeys + PAD	Followed by zero or two extra NULL for alignment on 32 bits

The events sent by the KeyboardArea to the owner application are:

Table 3.9.2-3 – KeyboardArea Event Structures: A661_EVT_KEY

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_KEY
A661 Key code of the pressed key	N/A	16	Key code

Keycodes are organized as following

The range 0x8000 – 0xFFFF is reserved for Implementation specific purpose

The range 0x0000 – 0x7FFF is standardized. The 16-bit are organized as following :

Table 3.9.2-4 – KeyboardArea Standard Keycodes

A661 Key code	Type	Size (bits)	Value/Description
Standard keycode marker	Bit	1	Set to 0, according to the reserved range for standardized keycodes.
OEMModifier4	Bit	1	If set, Implementation specific modifier 4 is pressed
OEMModifier3	Bit	1	If set, Implementation specific modifier 3 is pressed
OEMModifier2	Bit	1	If set, Implementation specific modifier 2 is pressed
OEMModifier1	Bit	1	If set, Implementation specific modifier 1 is pressed
Control key modifier flag	Bit	1	If set, indicates that Control key is pressed
Alt key modifier flag	Bit	1	If set, indicates that Alt key is pressed
Shift modifier flag	Bit	1	If set, indicates that Shift key is pressed
A661 standard Key code	uchar	8	ARINC739A Attachment 4 button push codes. Or one of the following A661 specific keycodes values : 0x09 : TAB key 0x0D : ENTER key 0x1B : ESC key 0x2C : Comma key 0x5C : Backslash key Any other values are reserved.

Available SetParameter identifiers and associated data structure are defined in Table 3.9.2-5.

Table 3.9.2-5 – KeyboardArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX,PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX,SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfKeys	uchar	8	A661_NUMBER_OF_KEYS
KeyArray[MaxNumberOfKeys]		{32}+	A661_KEY_ARRAY

3.9.3 ScrollWheelArea

Categories:

- Interactive

Description:
The ScrollWheelArea is transparent rectangular widget. The objective of this widget is to send Increment events from Crew member interactions (from scrollwheel interactions) in order to allow UA to simulate scrolling action at screen.

The principle is that this ScrollWheelArea widget will define an area inside which the scrollwheel events will be redirected to UA while the Cursor is over the area and when the scrollwheel input is not consumed by another widget.

Note: An input is considered consumed by a widget X when it is made unavailable for any widget to trigger internal behavior after being used by the aforesaid widget X.

The CDS will transmit the scrollwheel events as an increment value: Number of Increments. If the pilot scrolls quickly enough, the event transmission will occur up to CDS cycle.

ScrollWheelArea widgets that are drawn on top of other interactive widget do not prevent the widget underneath from generating events. Under these conditions, the ScrollWheelArea widget and the interactive widget underneath will still be able to generate events. This is one of the noted exceptions to the recommendation defined in Section 2.3.5.2 for overlapping widgets.

ScrollWheelArea Parameters are defined in Table 3.9.3-1.

Table 3.9.3-1 –ScrollWheelArea Parameters

Parameters	Change	Description
WidgetType	D	A661_SCROLL_WHEEL_AREA
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Enable	DR	Ability of the widget to be activated

Parameters	Change	Description
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget

ScrollWheelArea Creation Structure is defined in Table 3.9.3-2.

Table 3.9.3-2 –ScrollWheelArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value / Range when necessary
WidgetType	ushort	16	A661_SCROLL_WHEEL_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
UnusedPad	uchar	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	

The events sent by the ScrollWheelArea to the owner application are:

Table 3.9.3-3 –ScrollWheelArea Event Structures: A661_EVT_INCREMENT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_INCREMENT
UnusedPad	N/A	16	0
NbOfIncrements	long	32	Number of input device implementation dependent increments

Available SetParameter identifiers and associated data structure are defined in Table 3.9.3-4.

Table 3.9.3-4 –ScrollWheelArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulongx 2	32 x 2	A661_SIZE_XY

3.9.4 MapHorz_VertexBuffer

Categories:

- Map management

Description:
A MapHorz_VertexBuffer widget contains vertex data to be used by vertex-array based rendering of graphical primitives. MapItems may reference vertex data contained within these widgets to render geometry. Vertices are referenced either sequentially or via a sequence of indices.

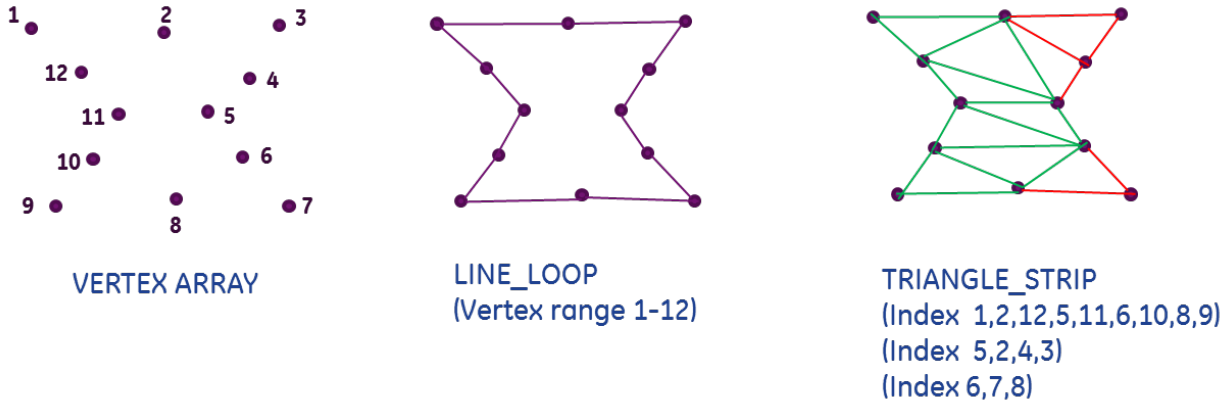


Figure 3.9.4-1 – Example Use Cases of the MapHorz_VertexBuffer

Subject to the overall limits of the MapHorz_VertexBuffer widget, the contents of the MapHorz_VertexBuffer may be replaced by setting parameters with new vertex data. A single command replaces the vertex data of the entire widget. The command is of variable length, but it always specifies the entire addressable length and contents of the widget.

When the map it drawn, the vertex data should be transformed as required to satisfy the MapSource projection.

Restriction:

A MapHorz_VertexBuffer must be in a MapHorz_Source container that is also the same container as any MapItemList items that reference it.

A MapHorz_Source contains a CDS defined fixed amount of memory allocation for MapHorz_VertexBuffer widgets. The number and maximum size of MapHorz_VertexBuffer widgets are established at definition time, and are not runtime modifiable. UAs use the set of MapHorz_VertexBuffer widgets to populate map data, using strategies such as having a static set of vertices that is not changed or swapping dynamic vertex data for updates using the MapSynchronizationNumber parameter.

MapHorz_VertexBuffer Parameters are defined in Table 3.9.4-1.

Table 3.9.4-1 – MapHorz_VertexBuffer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPHORZ_VERTEXBUFFER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Anonymous	D	Ability to be modified at run-time by the UA.
Specific parameters		
MaxNumberOfVertices	D	The maximum number of vertices to be defined for the widget.
NumberOfVertices	DR	The number of vertices currently defined for the widget.
BufferOfVertices	DR	The array of vertices as (X, Y) pairs.
MapSynchronizationNumber	R	See Section 3.2.8.4.

MapHorz_VertexBuffer Creation Structure is defined in Table 3.9.4-2.

Table 3.9.4-2 – MapHorz_VertexBuffer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPHORZ_VERTEXBUFFER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
MaxNumberOfVertices	ushort	16	Must be greater than or equal to 2
NumberOfVertices	ushort	16	Must be greater than or equal to 2, and must be less than or equal to MaxNumberOfVertices
Vertices	array of (long, long)	64 * NumberOfVertices	There are NumberOfVertices of (x, y) pairs

The MapHorz_VertexBuffer widget does not send any events.

Available SetParameter identifiers and associated data structure are defined in Table 3.9.4-3.

Table 3.9.4-3 – MapHorz_VertexBuffer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NumberOfVertices	ushort	16	A661_NUMBER_OF_ENTRIES
Vertices [up to MaxNumberOfVertices]	long x 2	{64}+	A661_VERTICES_ARRAY
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4

3.9.5 TouchArea

Categories:

- Graphical representation
- Interactive

Description:
The TouchArea widget, similar to CursorOver, allows the UA to be notified of touch events over the widget. A TouchArea widget is a rectangular widget whose graphical representation is implementation dependent.

This widget allows a UA to receive and interpret touch data without being limited to the CDS supported gesture types and gesture interpretation. If a UA requires a unique touch action or gesture for a particular interface, it can use the TouchArea widget to handle that function without requiring the CDS to implement a function only used by a single UA.

The touch IDs used in the TouchArea events must remain constant for a given touch point.

The AllowOutside parameter is used to dictate whether event data should continue to be sent by the CDS after that touch point moves outside of the widget active area. If a touch point starts outside of the widget active area, no events are sent for that touch point even after it moves into the active area. This is similar to the behavior of the CursorOver widget in that events are not sent until the widget area is entered.

This avoids issues with unresponsive or inadvertent activations that may occur if the UA would only start receiving event data part-way through a touch action.

If AllowOutside is A661_FALSE, CDS stops sending events after an A661_TOUCH_MOVE_OUTSIDE is generated, even if the touch point(s) move back inside the touch area or if the touch point(s) are lifted outside the area.

ReportMode is applied to each touch point in the event array. For example, if ReportMode is A661_REPORT_ON_TRANSITION, and a touch point is added, the event array only has one entry (the down State for the new touch point). Whereas if ReportMode is A661_REPORT_ALL and a touch point is added, the array will have an entry for each existing touch point, as well as the down State for the new touch point.

Generation of events among overlapping areas among multiple TouchArea widgets is implementation dependent.

TouchArea widgets that are drawn on top of other types of interactive widgets do not prevent the widgets underneath from generating events. Under these conditions, the TouchArea widget and the interactive widget underneath will both be able to generate events. This is one of the noted exceptions to the recommendation defined in Section 2.3.5.2 for overlapping widgets.

In Figure 3.9.5-1, a rectangle represents a touch area and the numbered circles represent touch points states along a single touch point input sequence. Filled amber circles correspond to events; dashed circles have no event.

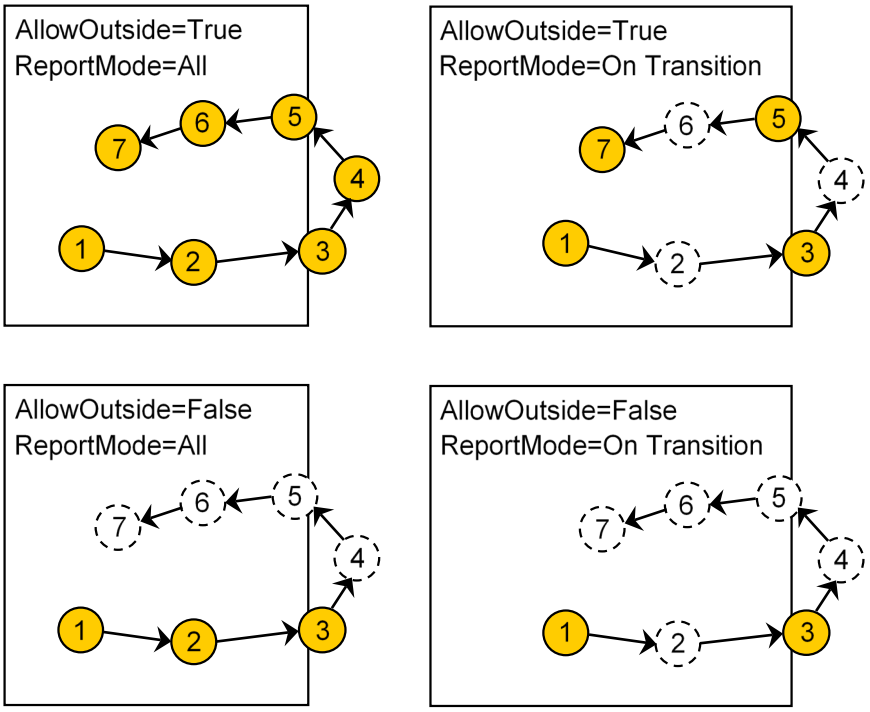


Figure 3.9.5-1 – TouchArea Event Generation Example

The touch State at each step in the above diagram is:

1. A661_TOUCH_DOWN
2. A661_TOUCH_MOVE
3. A661_TOUCH_MOVE_OUTSIDE
4. A661_TOUCH_MOVE_OUTSIDE
5. A661_TOUCH_MOVE
6. **A661_TOUCH_MOVE**
7. **A661_TOUCH_UP**

If AllowOutside=A661_FALSE, steps 4, 5, 6, and 7 do not generate events.

If ReportMode=A661_REPORT_ON_TRANSITION, steps 2, 4 and 6 do not generate events.

Restriction:
None

TouchArea Parameters are defined in Table 3.9.5-1.

Table 3.9.5-1 – TouchArea Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TOUCH_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 interactive
StyleSet	DR	Reference to the 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	DR	The width of the widget
SizeY	DR	The height of the widget

Parameters	Change	Description
Specific parameters		
ReportMode	D	Defines when events are generated from the widget. A661_REPORT_ON_TRANSITION - A661_EVT_TOUCH with State of A661_TOUCH_DOWN is sent when initial touch occurs. - A661_EVT_TOUCH with State of A661_TOUCH_UP is sent when a touch action ends. - A661_EVT_TOUCH with State of A661_TOUCH_MOVE_OUTSIDE is sent once as a touch exits the TouchArea. - A661_EVT_TOUCH with State of A661_EVT_TOUCH_MOVE is sent once as a touch re-enters the area (if AllowOutside is TRUE) A661_REPORT_ALL - In addition to A661_EVT_TOUCH with States of A661_TOUCH_DOWN and A661_TOUCH_UP, events with State of A661_TOUCH_MOVE (for touches inside the area) and A661_TOUCH_MOVE_OUTSIDE (for touches outside the area if AllowOutside is TRUE) are sent continuously while the touch action is in progress. In both cases: A661_EVT_TOUCH with State of A661_EVT_TOUCH_ABORT for touch actions which end abnormally; e.g., if the TouchArea becomes disabled or hidden, or the input is interrupted (the conditions in which such interruption may occur are implementation dependent). This state notifies the UA that the touch event sequence has stopped, such that it may maintain correct interaction state.
AllowOutside	D	Defines when events are generated from the widget when points move outside the widget active area. A touch point is required to start within the active area to report any events for that point. A661_FALSE When point(s) move outside the active area, send an A661_EVT_TOUCH event with State of A661_TOUCH_MOVE_OUTSIDE and stop sending events for those point(s). A661_TRUE Allow events to be generated when point(s) move outside the active area.
MaxTouchPoints	D	The maximum allowed simultaneous touch points **(must be greater than 0)**.
NumberOfTouchPoints	DR	Number of simultaneous touch points to be reported.

TouchArea Creation Structure is defined in Table 3.9.5-2.

Table 3.9.5-2 – TouchArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_TOUCH_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
UnusedPad	N/A	16	
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
ReportMode	uchar	8	A661_REPORT_ON_TRANSITION A661_REPORT_ALL
AllowOutside	uchar	8	A661_FALSE A661_TRUE
MaxTouchPoints	uchar	8	
NumberOfTouchPoints	uchar	8	

Available SetParameter identifiers and associated data structure are defined in Table 3.9.5-3.

Table 3.9.5-3 – TouchArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfTouchPoints	uchar	8	A661_NUMBER_OF_TOUCH_POINTS

The events sent by TouchArea to the owner application are defined in Table 3.9.5-4.

Table 3.9.5-4 – TouchArea Event: A661_EVT_TOUCH

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_TOUCH
TouchPointsReported	uchar	8	
UnusedPad	N/A	8	
TouchIDs[TouchPointsReported]	{uchar}	8 * TouchPointsReported + Pad	Touch ID of the touch point
TouchState[TouchPointsReported]	{uchar}	8 * TouchPointsReported + Pad	State of the touch point A661_TOUCH_DOWN A661_TOUCH_MOVE A661_TOUCH_MOVE_OUTSIDE A661_TOUCH_UP A661_TOUCH_ABORT
RelPosX[TouchPointsReported]	{long}	32 * TouchPointsReported	Relative X position of the touch point as an offset to the widget's origin.
RelPosY[TouchPointsReported]	{long}	32 * TouchPointsReported	Relative Y position of the touch point as an offset to the widget's origin.
Timestamp[TouchPointsReported]	{long}	32 * TouchPointsReported	**Time stamp for the touch point data, in microseconds. It increases until the maximum value is reached, at which point the value wraps around to the minimum value. Its origin/reference should be consistent across widgets, but is otherwise meaningless since its purpose is to calculate differences in time. Its precision and granularity are implementation dependent.** If UAs need to be able to compute velocity and acceleration, the timestamp should accommodate this capability.

3.9.6 GestureArea

Categories:

- Graphical representation
- Interactive

Description:
The GestureArea widget, similar to CursorOver, allows a UA to be notified of gesture events generated from user interaction over the widget area. A GestureArea widget is a rectangular widget whose graphical representation is implementation dependent.

This widget allows a UA to receive gesture data with minimal interpretation. Unlike the TouchArea widget, in which the UA receives basic touch information and must determine how to interpret it, with a GestureArea the CDS performs the touch data interpretation and passes on the end (or when appropriate, intermediate) results in discrete events.

The AllowOutside parameter is used to dictate whether event data should continue to be sent by the CDS after a touch point moves outside of the widget active area. If AllowOutside is FALSE and any touches move outside the area, the gesture is aborted. If a touch point starts outside of the widget active area, no events are sent for that touch point even after it moves into the active area. This is similar to the behavior of the CursorOver widget in that events are not sent until the widget area is entered. This avoids issues with unresponsive or inadvertent activations that may occur if the UA would only start receiving event data part-way through a gesture.

Generation of events among overlapping areas of multiple GestureArea widgets is implementation dependent.

GestureArea widgets that are drawn on top of other types of interactive widgets do not prevent the widgets underneath from generating events. Under these conditions, the GestureArea widget and the interactive widget underneath will both be able to generate events. This is one of the noted exceptions to the recommendation defined in Section 2.3.5.2 for overlapping widgets.

Distinguishing between gestures, especially at the start of a touch action, may be complicated. Arbitration between gestures is implementation dependent. Therefore, the CDS must determine when to report which events, such that the UA does not receive ambiguous information. If a UA needs to perform its own arbitration, the TouchArea widget should be used instead. The purpose of the GestureArea widget is to provide UAs with a way to receive gesture events abstracted from interpretation logic; the GestureArea events are determined by CDS in a uniform manner in an effort to achieve system-wide commonality across the user interface.

For example, at the start of a touch action, it may not be immediately evident whether the gesture being performed is a (single) tap or the start of a double-tap. CDS may need to wait for a double-tap to fail before sending out a (single) tap event. Similarly, distinguishing between a flick or drag may require a certain duration of time to assess velocity or direction information before CDS can report the appropriate event data.

Restriction:
None

COMMENTARY

UA developers should consider the implications of implementation dependent gesture arbitration in their design. Latency, visual feedback, and user error/response are all some of the elements that may potentially affect the overall system implementation. Accounting for these factors in the development of the touch interface is the responsibility of both CDS and UA suppliers.

GestureArea Report States are defined in Table 3.9.6-1.

Table 3.9.6-1 – GestureArea Report States

Gesture State	Description
A661_GESTURE_CANDIDATE	Sent when implementation dependent rules decide that it may be possible initial match for a gesture (number or points, location, separation of points or other criteria) This will occur before a gesture starts to be fully recognized as a gesture. A gesture event sequence containing a CANDIDATE should be followed by an event of START or ABORT. Report may generate more than 1 CANDIDATE for same gesture but is implementation dependent
A661_GESTURE_START	Sent once when the gesture is recognized as a gesture. A gesture event sequence containing a START should end with an event having State of END or ABORT.
A661_GESTURE_UPDATE	Sent continuously while the gesture is in progress only after a GESTURE_START has been already sent.
A661_GESTURE_END	Sent when a gesture is complete.
A661_GESTURE_ABORT	Sent once when gesture ends before completing; e.g. if the GestureArea becomes disabled or hidden, if the gesture is aborted because a point exited the area and AllowOutside is FALSE, or the input is interrupted (the conditions in which such interruption may occur are implementation dependent). An ABORT will only be sent after a CANDIDATE or START has already been sent. This state notifies the UA that the gesture event sequence has stopped, such that it may maintain correct interaction states.

GestureArea Report Modes and the gesture States reported are defined in Table 3.9.6-2.

Table 3.9.6-2 – GestureArea Report Modes

Report Mode \ Gesture State	A661_GESTURE_CANDIDATE	A661_GESTURE_START	A661_GESTURE_UPDATE	A661_GESTURE_END	A661_GESTURE_ABORT
A661_REPORT_ON_COMPLETION				X	
A661_REPORT_ON_TRANSITION		X		X	X
A661_REPORT_ON_TRANSITION_WITH_CANDIDATE	X	X		X	X
A661_REPORT_ALL		X	X	X	X
A661_REPORT_ALL_WITH_CANDIDATE	X	X	X	X	X

GestureArea Parameters are defined in Table 3.9.6-3.

Table 3.9.6-3 – GestureArea Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_GESTURE_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 interactive
StyleSet	DR	Reference to the 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	DR	The width of the widget
SizeY	DR	The height of the widget
Specific parameters		
ReportMode	D	Defines when events are generated from the widget. A661_REPORT_ON_COMPLETION A661_REPORT_ON_TRANSITION A661_REPORT_ALL **A661_REPORT_ON_TRANSITION_WITH_CANDIDATE** **A661_REPORT_ALL_WITH_CANDIDATE**
AllowOutside	D	Defines when events are generated from the widget when one or more points associated with an ongoing gesture move outside the widget active area. A661_FALSE Disallow events when point(s) move outside the active area. The gesture interaction is restricted to the active area. A661_TRUE Allow events to be generated when point(s) move outside the active area. The gesture is said to continue beyond the widget area.
NumberOfEvents	D	The number of events in the following EventIdArray

Parameters	Change	Description
EventIdArray	D	Array to define which events the UA will receive using the enumerations below. See Section 4 for details about the gestures and associated events. A661_GESTURE_TAP - A661_EVT_GESTURE_TAP A661_GESTURE_DOUBLE_TAP - A661_EVT_GESTURE_DOUBLE_TAP A661_GESTURE_PRESS_AND_HOLD - A661_EVT_GESTURE_PRESS_AND_HOLD **A661_GESTURE_HOLD** - **A661_EVT_GESTURE_HOLD** A661_GESTURE_PINCH - A661_EVT_GESTURE_PINCH A661_GESTURE_ROTATE - A661_EVT_GESTURE_ROTATE A661_GESTURE_DRAG - A661_EVT_GESTURE_DRAG A661_GESTURE_FLICK - A661_EVT_GESTURE_FLICK A661_GESTURE_FLICK_LEFT - A661_EVT_GESTURE_DIR_FLICK with Direction = A661_LEFT A661_GESTURE_FLICK_RIGHT - A661_EVT_GESTURE_DIR_FLICK with Direction = A661_RIGHT A661_GESTURE_FLICK_UP - A661_EVT_GESTURE_DIR_FLICK with Direction = A661_UP A661_GESTURE_FLICK_DOWN - A661_EVT_GESTURE_DIR_FLICK with Direction = A661_DOWN

GestureArea Creation Structure is defined in Table 3.9.6-4.

Table 3.9.6-4 – GestureArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_GESTURE_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
Visible	uchar	8	A661_FALSE A661_TRUE
StyleSet	ushort	16	
UnusedPad	N/A	16	
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
ReportMode	uchar	8	A661_REPORT_ON_COMPLETION A661_REPORT_ON_TRANSITION A661_REPORT_ALL **A661_REPORT_ON_TRANSITION_WITH_CANDIDATE** **A661_REPORT_ALL_WITH_CANDIDATE**
AllowOutside	uchar	8	A661_FALSE A661_TRUE
NumberOfEvents	uchar	8	
UnusedPad	N/A	8	0
EventIdArray[NumberOfEvents]	{**uchar**}+	**8** * NumberOfEvents + Pad	There are NumberOfEvents in the array. The complete array is followed by **zero, one, two, or three bytes(s) of padding to be 32 bits aligned**.

Available SetParameter identifiers and associated data structure are defined in Table 3.9.6-5.

Table 3.9.6-5 – GestureArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY

The events sent by GestureArea to the owner application are defined in Section 3.9.6.1.

3.9.6.1 Gesture Library

3.9.6.1.1 A661_GESTURE_TAP

The tap gesture will generate an A661_EVT_GESTURE_TAP event when a point down is followed by an immediate lift off.

- Action: A point down followed by an immediate lift off
- One or more points
- Reports after completion

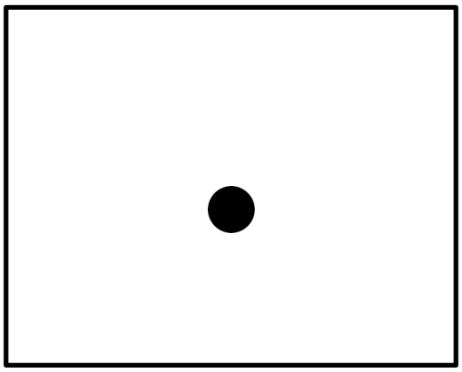


Figure 3.9.6-1 – Single Point Tap

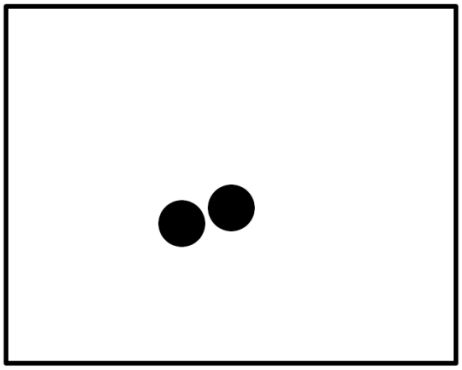


Figure 3.9.6-2 – Two Point Tap

Table 3.9.6.1.1-1 – Gesture Event: A661_EVT_GESTURE_TAP

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_TAP
NumberOfPoints	uchar	8	Number of current points used in the gesture.
UnusedPad	N/A	8	0
RelPosX	long	32	Relative X position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.

3.9.6.1.2 A661_GESTURE_DOUBLE_TAP

The double tap gesture will generate an A661_EVT_GESTURE_DOUBLE_TAP event after two taps in quick succession in close proximity occur.

- Action: Two taps in quick succession in close proximity
- One or more points
- Reports after completion

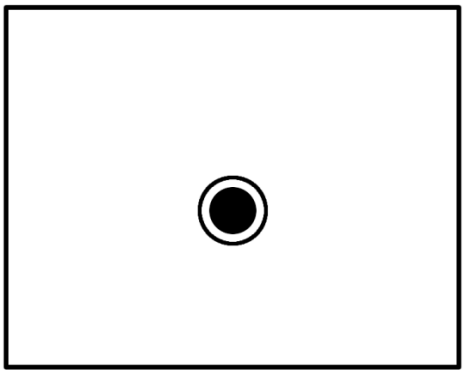


Figure 3.9.6-3 – Single Point Double Tap

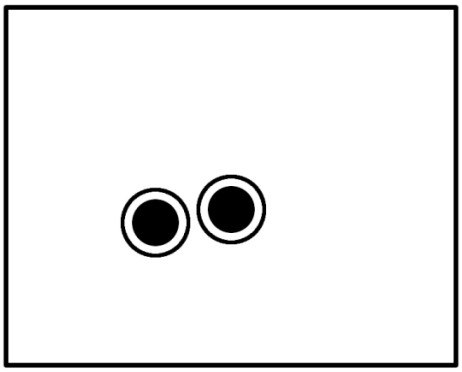


Figure 3.9.6-4 – Two Point Double Tap

Table 3.9.6.1.2-1 – Gesture Event: A661_EVT_GESTURE_DOUBLE_TAP

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_DOUBLE_TAP
NumberOfPoints	uchar	8	Number of current points used in the gesture.
UnusedPad	N/A	8	0
RelPosX	long	32	Relative X position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.

3.9.6.1.3 A661_GESTURE_PRESS_AND_HOLD

The gesture begins when one or more points go down, which results in an A661_EVT_GESTURE_PRESS_AND_HOLD event with a State of A661_GESTURE_START being generated. When the point(s) have been held in close proximity to their start location for a duration, the gesture completes and an A661_EVT_GESTURE_PRESS_AND_HOLD event with a State of A661_GESTURE_END is sent.

A start duration can be used to differentiate a press and hold event from a tap event.

- Action: Point(s) down and held, in close proximity, for a duration
- One or more points
- Reports on start and end

When gesture candidate reporting is enabled the press and hold gesture will report when implementation dependent criteria for candidate status are met. At this time the A661_EVT_GESTURE_PRESS_AND_HOLD event with a State of A661_CANDIDATE will be generated

Once the gesture is recognized an A661_EVT_GESTURE_PRESS_AND_HOLD event with a State of A661_GESTURE_START will be generated

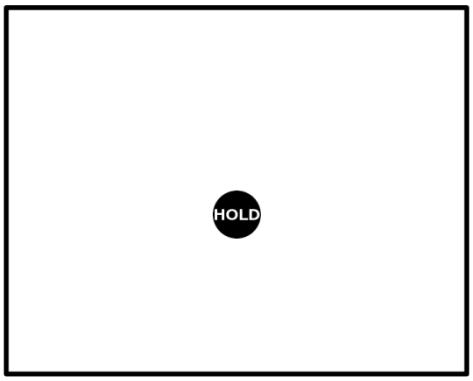


Figure 3.9.6-5 – Press and Hold

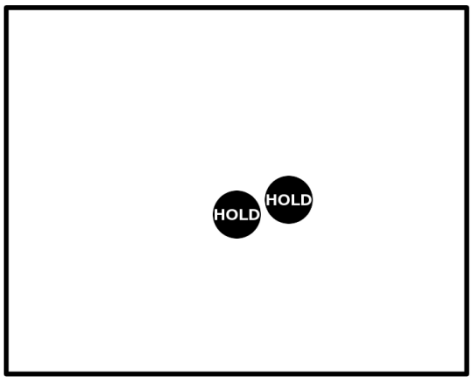


Figure 3.9.6-6 – Two Point Press and Hold

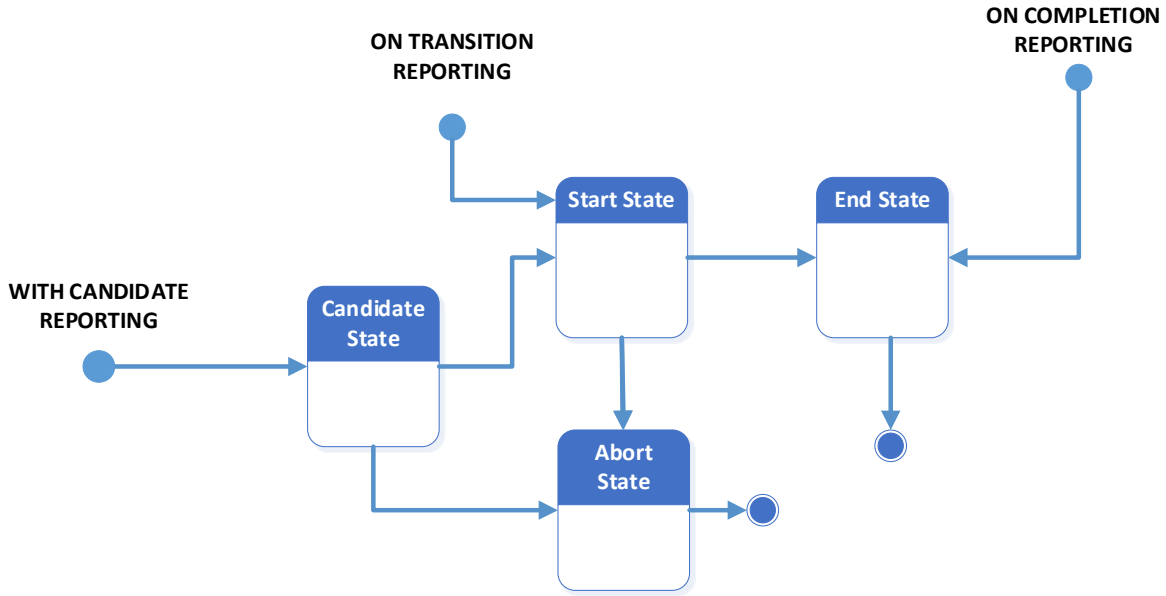


Figure 3.9.6-7 – Press and Hold Reporting Mode State Transition

Table 3.9.6.1.3-1 – Gesture Event: A661_EVT_GESTURE_PRESS_AND_HOLD

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_PRESS_AND_HOLD
NumberOfPoints	uchar	8	Number of current points used in the gesture.
State	uchar	8	**A661_GESTURE_CANDIDATE** A661_GESTURE_START A661_GESTURE_END A661_GESTURE_ABORT
RelPosX	long	32	Relative X position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.

3.9.6.1.4 A661_GESTURE_PINCH

The pinch expand gesture will report events continuously while the gesture is ongoing. The gesture begins when two points go down and start to move away or toward each other which results in an A661_EVT_GESTURE_PINCH event with a State of A661_GESTURE_START being generated. While the points are down A661_EVT_GESTURE_PINCH events with a State of A661_GESTURE_UPDATE are generated when the calculated scale is changed. When the points are removed the gesture completes and an A661_EVT_GESTURE_PINCH event with a State of A661_GESTURE_END is sent.

- Action: Two points move toward or away from each other
- Reports continuously as scale changes

When Gesture candidate reporting is enabled the drag gesture will report when implementation dependent criteria for candidate status are met. At this time the A661_EVT_GESTURE_PINCH event with a State of A661_CANDIDATE will be generated

Once the gesture is recognized by the points moving away from the initial down position an A661_EVT_GESTURE_PINCH event with a State of A661_GESTURE_START will be generated.

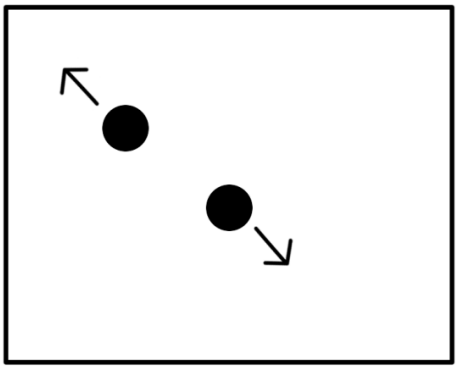


Figure 3.9.6-8 – Pinch or Expand Two Point Toward or Away From Each Other

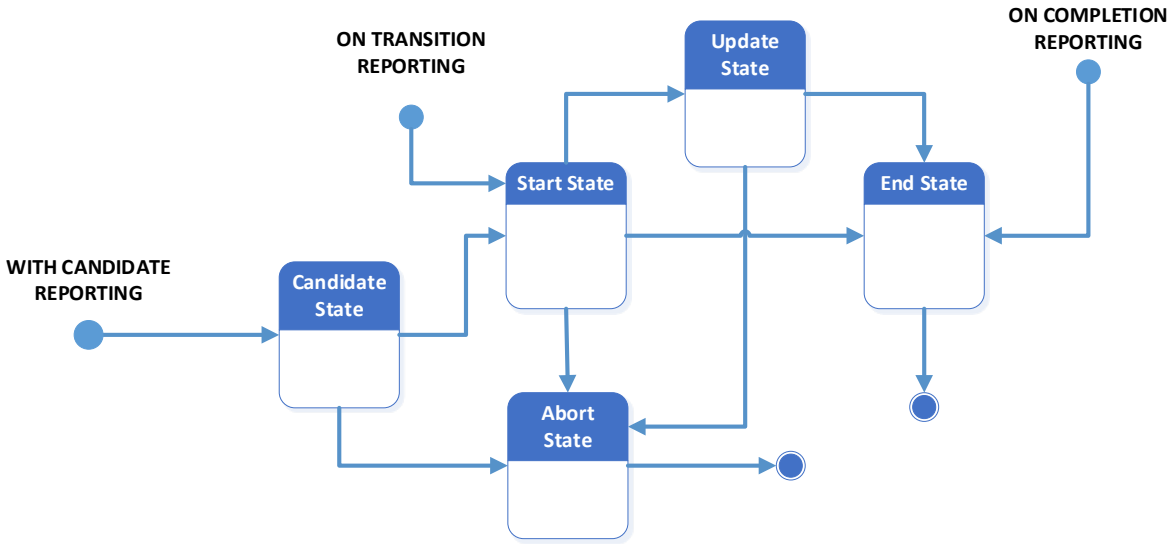


Figure 3.9.6-9 – Pinch Reporting Mode State Transition

Table 3.9.6.1.4-1 – Gesture Event: A661_EVT_GESTURE_PINCH

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_PINCH
State	uchar	8	**A661_GESTURE_CANDIDATE** A661_GESTURE_START A661_GESTURE_UPDATE A661_GESTURE_END A661_GESTURE_ABORT
UnusedPad	N/A	8	0
RelPosX	long	32	Relative X position of the midpoint between the two touch points as an offset to the widget's origin.
RelPosY	long	32	Relative Y position of the midpoint between the two touch points as an offset to the widget's origin.
Scale	float	32	Ratio of the current distance between points to the start distance between points. For example, if the initial distance between points is 1 cm, and the current distance between points is 3.2 cm, the Scale value is 3.2. When State is A661_GESTURE_START, Scale is 1.0.

3.9.6.1.5 A661_GESTURE_ROTATE

The rotate gesture will report events continuously while the gesture is ongoing. The gesture begins when two points go down and start to move in a clockwise or counter-clockwise direction which results in an A661_EVT_GESTURE_ROTATE event with a State of A661_GESTURE_START being generated. While the points are down A661_EVT_GESTURE_ROTATE events with a State of A661_GESTURE_UPDATE are generated when the calculated angle is changed. When the points are removed the gesture completes and an A661_EVT_GESTURE_ROTATE event with a State of A661_GESTURE_END is sent.

- Action: Two points with one stationary and one moving in an arc around the other. Or, two points both moving in the same rotation direction around a virtual mid-point in between them
- Reports continuously as angle or midpoint changes

When gesture candidate reporting is enabled the rotate gesture will report as when implementation dependent criteria for candidate status are met. At this time the A661_EVT_GESTURE_ROTATE event with a State of A661_CANDIDATE will be generated

Once the gesture is recognized by the points moving clockwise or counter clockwise direction an A661_EVT_GESTURE_ROTATE event with a State of A661_GESTURE_START will be generated

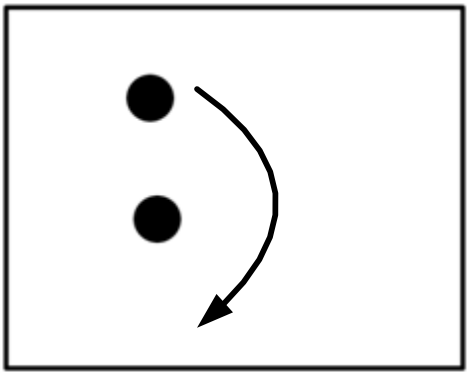


Figure 3.9.6-10 – Rotate One Point About Another Point

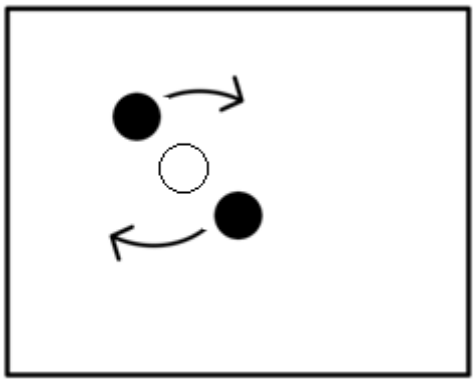


Figure 3.9.6-11 – Rotate Two Points About a Center

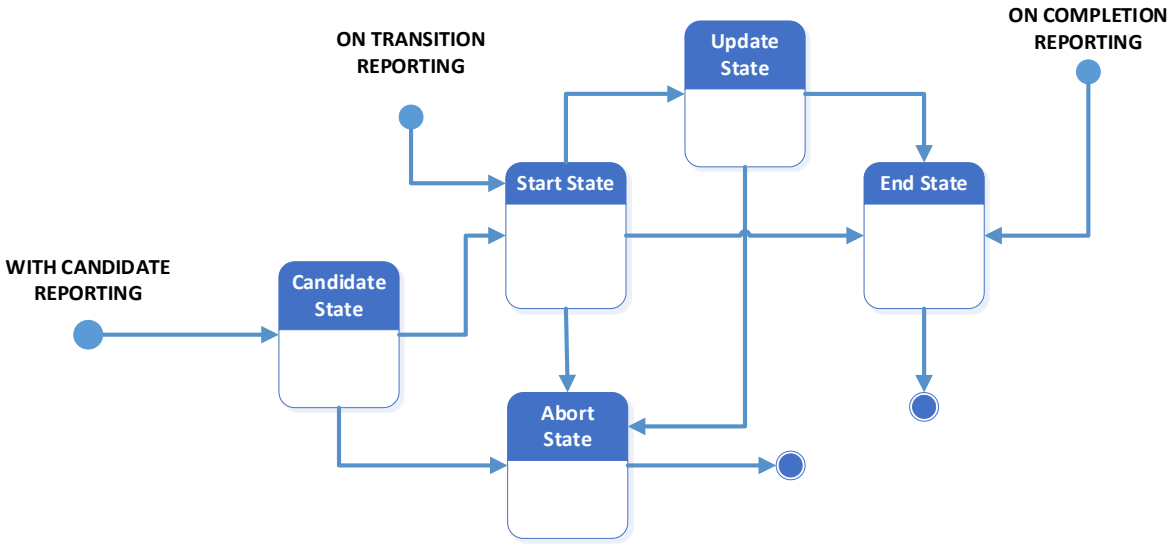


Figure 3.9.6-12 – Rotate Reporting Mode State Transition

Table 3.9.6.1.5-1 – Gesture Event: A661_EVT_GESTURE_ROTATE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_ROTATE
State	uchar	8	**A661_GESTURE_CANDIDATE** A661_GESTURE_START A661_GESTURE_UPDATE A661_GESTURE_END A661_GESTURE_ABORT
UnusedPad	N/A	8	0
RelPosX	long	32	Relative X position of the reference point of the action as an offset to the widget's origin. The determination of the reference point is implementation dependent.
RelPosY	long	32	Relative Y position of the reference point of the action as an offset to the widget's origin. The determination of the reference point is implementation dependent.
RelAngle	float	32	Angle of rotation, in degrees, of two or more points from the starting point to the current point. Positive angles indicate counter-clockwise rotation.

3.9.6.1.6 A661_GESTURE_DRAG

The drag gesture will report events continuously while the gesture is ongoing. The gesture begins when one, or more, points go down and start to move away from the initial down position which results in an A661_EVT_GESTURE_DRAG event with a State of A661_GESTURE_START being generated. While the point(s) are down, A661_EVT_GESTURE_DRAG events with a State of A661_GESTURE_UPDATE are generated when the point(s) position is changed. When the point(s) are removed the gesture completes and an A661_EVT_GESTURE_DRAG event with a State of A661_GESTURE_END is sent.

- Action: Point down followed by motion away from the initial position
- One or more points
- Reports continuously as X or Y position changes

When gesture candidate reporting is enabled the drag gesture will report when implementation dependent criteria for candidate status are met. At this time the A661_EVT_GESTURE_DRAG event with a State of A661_CANDIDATE will be generated

Once the gesture is recognized by the points moving away from the initial down position an A661_EVT_GESTURE_DRAG event with a State of A661_GESTURE_START will be generated.

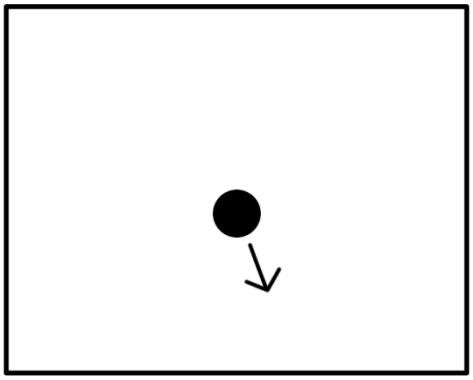


Figure 3.9.6-13 – Single Point Drag

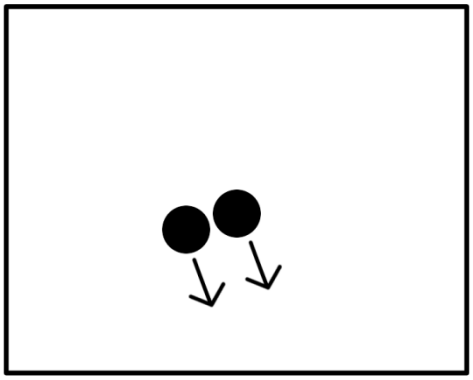


Figure 3.9.6-14 – Two Point Drag

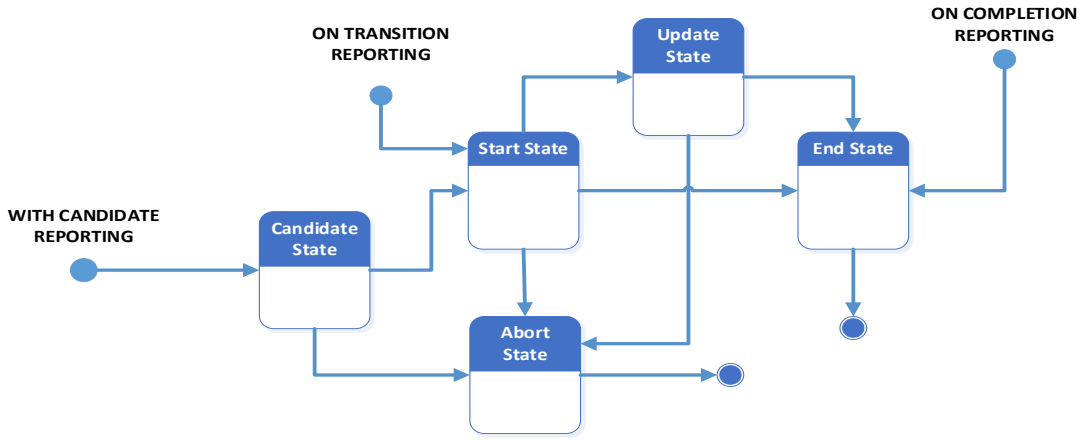


Figure 3.9.6-15 – Drag Reporting Mode State Transition

Table 3.9.6.1.6-1 – Gesture Event: A661_EVT_GESTURE_DRAG

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_DRAG
NumberOfPoints	uchar	8	Number of current points used in the gesture.
State	uchar	8	A661_GESTURE_CANDIDATE A661_GESTURE_START A661_GESTURE_UPDATE A661_GESTURE_END A661_GESTURE_ABORT
RelPosX	long	32	Relative X position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.

3.9.6.1.7 A661_GESTURE_FLICK

The flick gesture generates an A661_EVT_GESTURE_FLICK event when one or more points go down and then quickly move away and lift off. The event reports the velocity in the X and Y dimensions.

The calculated velocity is in units of 1/100 of millimeter per second.

- Action: Point down followed by a quick motion away from the initial point
- One or more points
- Reports after completion

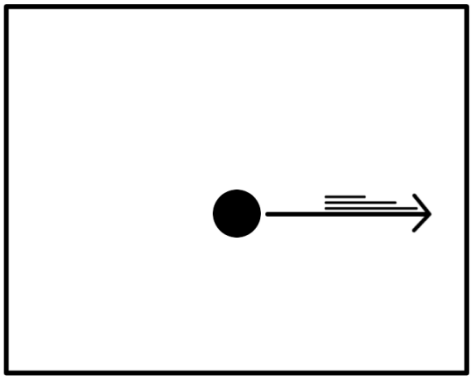


Figure 3.9.6-16 – Single Point Flick

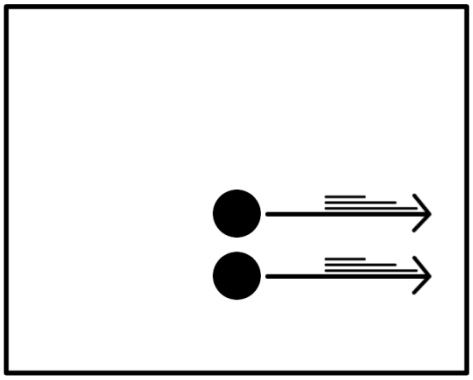


Figure 3.9.6-17 – Two Point Flick

Table 3.9.6.1.7-1 – Gesture Event: A661_EVT_GESTURE_FLICK

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_FLICK
NumberOfPoints	uchar	8	Number of current points used in the gesture.
UnusedPad	N/A	8	0
RelPosX	long	32	Relative X position of the start point of the gesture as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the start point of the gesture as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
VelocityX	float	32	Velocity of gesture in the X direction, given in units of 1/100 of millimeter per second.
VelocityY	float	32	Velocity of gesture in the Y direction, given in units of 1/100 of millimeter per second.

3.9.6.1.8 A661_GESTURE_DIR_FLICK

The flick gesture generates an A661_EVT_GESTURE_DIR_FLICK event when one, or more, points go down and then quickly move away and lift off. The event reports the velocity in the X and Y dimensioned.

The calculated velocity is in units of 1/100 of millimeter per second.

- Action: Point down followed by a quick motion away from the initial point
- One or more points
- Reports after completion

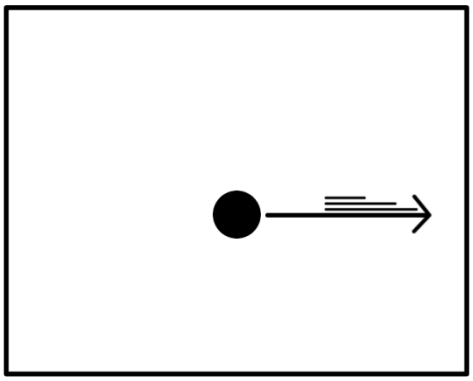


Figure 3.9.6-18 – Single Point Directional Flick

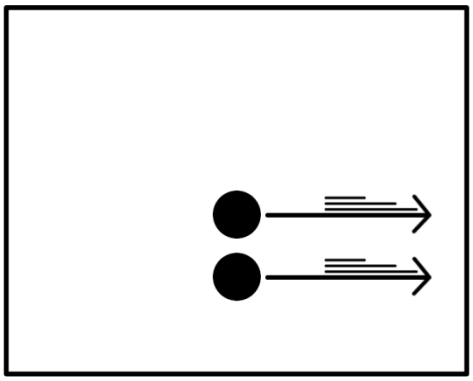


Figure 3.9.6-19 – Two Point Directional Flick

Table 3.9.6.1.8-1 – Gesture Event: A661_EVT_GESTURE_DIR_FLICK

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_DIR_FLICK
NumberOfPoints	uchar	8	Number of current points used in the gesture.
Direction	N/A	8	A661_LEFT A661_RIGHT A661_UP A661_DOWN
RelPosX	long	32	Relative X position of the start point of the gesture as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the start point of the gesture as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
Velocity	float	32	Velocity of gesture in the direction of the flick, given in units of 1/100 of millimeter per second.

3.9.6.1.9 A661_GESTURE_HOLD

The gesture begins when one or more points go down, which results in an A661_EVT_GESTURE_HOLD event with a State of A661_GESTURE_START being generated. When the point(s) have been held in close proximity to their start location for a required duration, an A661_EVT_GESTURE_HOLD event with a State of A661_GESTURE_UPDATE is sent to indicate the duration has been satisfied. While the touch point(s) are still down, A661_GESTURE_UPDATE events continue to be sent with an indication of the duration of the gesture since the start event.

When the point(s) are removed, an A661_EVT_GESTURE_HOLD event with a State of A661_GESTURE_END is sent. If the duration criteria was not met before the point(s) are removed, an A661_EVT_GESTURE_HOLD event with a State of A661_GESTURE_ABORT is sent.

The A661_EVT_GESTURE_HOLD gesture is differentiated from A661_EVT_GESTURE_PRESS_AND_HOLD in that it will continue to provide events as long as the touch point(s) are held down, and will also provide the duration of the hold in the event data.

- Action: Point(s) down and held, in close proximity, for a duration
- One or more points
- Reports on start, update (once hold duration reached), and end

When Gesture candidate reporting is enabled the Hold gesture will report when implementation dependent criteria for candidate status are met. At this time the A661_EVT_GESTURE_HOLD event with a State of A661_CANDIDATE will be generated

Once the gesture is recognized an A661_EVT_GESTURE_HOLD event with a State of A661_GESTURE_START will be generated

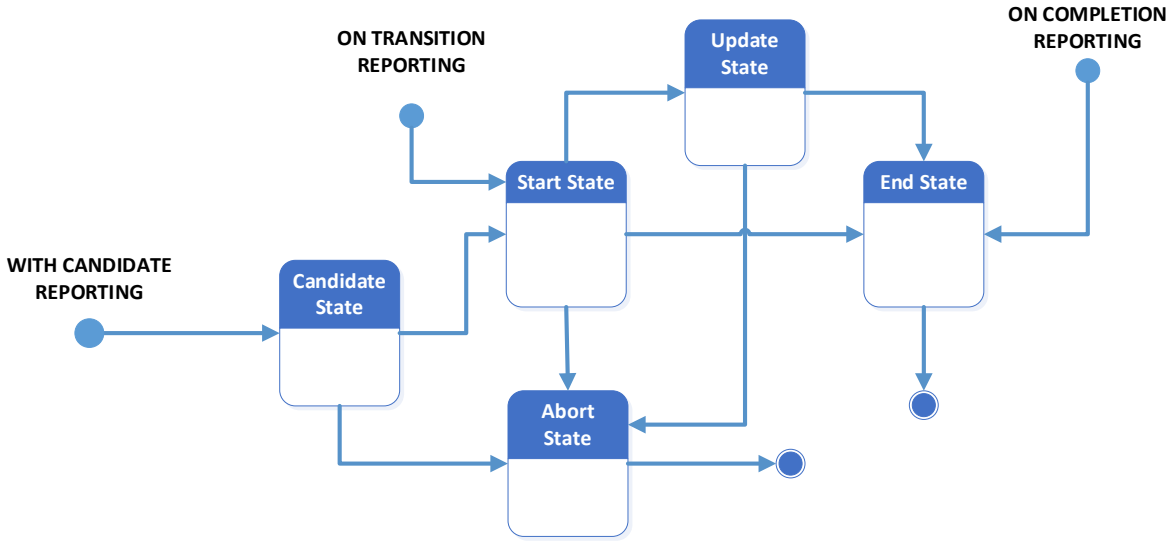


Figure 3.9.6-20 –Hold Reporting Mode State Transition

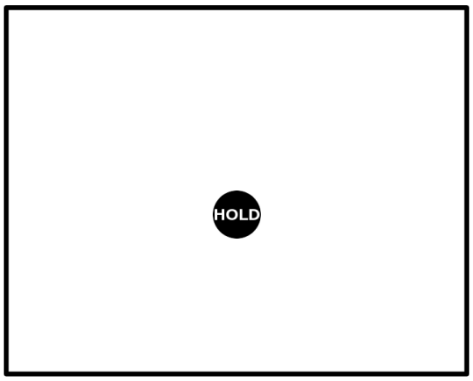


Figure 3.9.6-21 –Hold

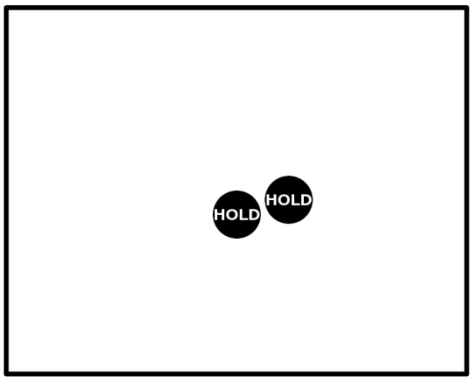


Figure 3.9.6-22 – Two Point Hold

Table 3.9.6.1.9 -1 – Gesture Event: A661_EVT_GESTURE_HOLD

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_GESTURE_HOLD
NumberOfPoints	uchar	8	Number of current points used in the gesture.
State	uchar	8	A661_GESTURE_CANDIDATE A661_GESTURE_START A661_GESTURE_UPDATE A661_GESTURE_END A661_GESTURE_ABORT
RelPosX	long	32	Relative X position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
RelPosY	long	32	Relative Y position of the gesture action as an offset to the widget's origin. The position reported when NumberOfPoints > 1 is implementation dependent.
Duration	long	32	Time that the touch point(s) have been held down since A661_GESTURE_START, in microseconds.

3.9.7 InkArea

Categories:

- Graphical representation
- Interactive

Description:
The InkArea widget allows a crew member to draw in an active area region of the widget using input from a digitizer or similar device. This widget provides a means collect natural written input data on a canvas defined by the UA upon which the CDS can draw based on crew input.

Impact to drawn content when PosX, PosY, SizeX, or SizeY is changed is implementation dependent.

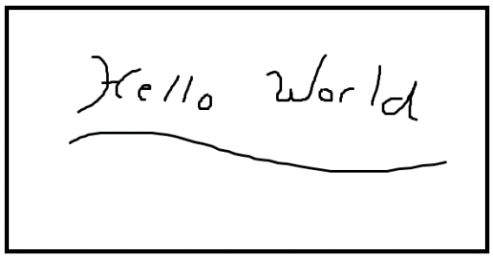


Figure 3.9.7-1 – User Draws "Hello World" in the Ink Area Widget

Restriction:
None

InkArea Parameters are defined in Table 3.9.7-1.

Table 3.9.7-1 – InkArea Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_INK_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 interactive
StyleSet	DR	Reference to the 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	DR	The width of the widget
SizeY	DR	The height of the widget
Specific parameters		
ColorIndex	DR	Color index of the ink graphic. See Section 3.1.3.3.1.
Filled	DR	If set to True, background will be filled
FillIndex	DR	Fill Pattern index for the background. See Section 3.1.3.3.1.
Clear	R	Set to A661_TRUE to clear the ink area. CDS will set this value back to A661_FALSE.
MaxBufferSize	D	Implementation dependent value to set max size of ink data
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.

InkArea Creation Structure is defined in Table 3.9.7-2.

Table 3.9.7-2 – InkArea Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_INK_AREA
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE **A661_TRUE_WITH_VALIDATION**
StyleSet	ushort	16	
UnusedPad	N/A	16	
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
ColorIndex	uchar	8	(valid palette index)
Filled	uchar	8	A661_FALSE A661_TRUE
FillIndex	uchar	8	(valid Fill Pattern index)
UnusedPad	N/A	8	
MaxBufferSize	ushort	16	
NextFocusedWidget	ushort	16	

Available SetParameter identifiers and associated data structure are defined in Table 3.9.7-3.

Table 3.9.7-3 – InkArea Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX,SizeY	ulong x 2	32 x 2	A661_SIZE_XY
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Clear	uchar	8	A661_CLEAR
NextFocusedWidget	ushort	16	A661_NEXT_FOCUSED_WIDGET
Filled	**uchar**	**8**	**A661_FILLED**

The InkArea widget does not send any event.

3.9.8 AnimationOnParam

Categories:

- Utility

Description:
The objective of this widget is to provide a means for performing an animation on one widget and a parameter.

- If the TargetParamID value correspond to a runtime parameter type specified for the TargetWidgetID widget type (see Section 4.6.8), the parameter to animate is specified by the corresponding parameter type (e.g., 0xB1E0 will animate the A661_FILL_INDEX for the GpRectangle). Note that parameters which do not correspond to a graphical characteristic (for example, the A661_ENABLE parameter for a PushButton) cannot be animated.
- If the TargetParamID value is in the range reserved for OEM customization (0xB800 – 0xBFFF), the parameter to animate is implementation dependent. As an example, its value can be used to specify the corresponding widgetCapacities attribute defined in the LookCapacities (see Appendix J), or an attribute defined for a component used in this widget.

Note that the list of parameters which can be animated for each widget is implementation dependent.

The widget referenced in this AnimationOnParam widget must be defined in the Definition File before the AnimationOnParam widget.

The behavior for this widget is specified in Section 3.2.11.

If an animation starts for a widget and a parameter, and a second one starts for the same widget and parameter before the first one has finished, the result is implementation dependent.

Case where TargetParamID specifies a runtime parameter:

The value of the TargetParamID parameter specifies the ParameterType of the parameter to animate. For example, if we want to animate the fill color of a GpRectangle:

- The 0xB1E0 value (which is the value of the A661_FILL_INDEX parameter) specifies that we want to set an animation changing the fillColor of the GpRectangle.

The animation will animate the value of the target parameter from FromValue to ToValue in the specified Duration. For example, for the above example, if the index 0x01 correspond to the color Red, and the index 0x05 to the color Green, then if FromValue=0x01, and ToValue = 0x05, the animation will animate the color from Red to Green.

Example of implementation dependent case where TargetParamID specifies a widgetCapacities attribute:

The value of the TargetParamID parameter specifies the Attribute of the Widget Capacities to animate. This value must correspond to an attribute defined for the widget in the LookCapacities, with the specified “**id**” XML attribute (or an attribute defined for a component used in this widget). For example, if we want to animate

the Stroke width of a GpLine, suppose we have the following specification for the LookCapacities for the A661_GP_LINE widget:

```
<attributes>
  <attribute name="LineWidth" typeRef="float" scope="styleset"
        id=”0xB8F1/>
  …
</attributes>
```

The 0xB8F1 value (which is the value of the LineWidth attribute) specifies that we want to set an animation changing the line width of the GpLine.

Notes:

1. The effective type of FromValue and ToValue depend on the type of the parameter specified by the TargetParamID value. For example, in the first example (specifying the A661_FILL_INDEX parameter), the type of the values will be uchar. In the second example, (the Stroke width), the type of the values will be float. If the size of the type is smaller than 32-bit, then FromValue and ToValue will be padded by 0s. The padding is performed at the end of the effective value.
2. **The CDS ensures that the change from the “FromValue” to the “ToValue” value is continuous, regardless of the effective type of the parameter to animate**
3. If the animation is applied on a runtime parameter, then the value of the parameter is modified by the animation. For example, if an AnimationOnParam is applied to the A661_POS_X of a GpRectangle with a PosX position of 1000, and ToValue = 4000, the graphical position of the GpRectangle at the end of the animation will be 4000. If **the animation is in progress**, then the animation takes precedence over the runtime modification of the parameter by the UA.
4. If the animation is applied on a LookCapacities attribute, then the animation takes precedence over the StyleSet modification of the parameter by the UA.
5. **If several animations apply on the same parameter and are in progress at the same time, the result is implementation dependent.**
6. **If this animation receives a A661_DEACTIVATE, it will just stop the animation (no more update is applied on the target parameter value).**

AnimationOnParam Parameters are defined in Table 3.9.8-1.

Table 3.9.8-1 – AnimationOnParam Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ANIMATION_ONPARAM
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
StyleSet	DR	Reference to predefined graphical characteristics inside CDS

Parameters	Change	Description
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
TargetWidgetID	D	Widget ID which should be animated according to this animation
TargetParamID	D	The ParameterType of the parameter to animate (see Section 4.6.8)
FromValue	DR	The initial value of the animation. The transformation from the value to the associated parameter to animate is implementation dependent
ToValue	DR	The final value of the animation. The transformation from the value to the associated parameter to animate is implementation dependent.
AnimationLawRef	DR	The animation law reference to use for the animation (see Section 9.0)
Duration	DR	The duration of the animation in milliseconds
Delay	DR	The delay in milliseconds for starting the animation
AnimationState	R	The state of the animation A661_PLAY A661_ABORT A661_END **A661_DEACTIVATE** **A661_FROM** **A661_TO**
AnimationRepetition	DR	Specifies how much times the animation will repeat (0 or 1 will mean no repetition, and the maximum value for ushort an indefinite repetition of the animation)

AnimationOnParam Creation Structure is defined in Table 3.9.8-2.

Table 3.9.8-2 – AnimationOnParam Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ANIMATION_ONPARAM
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
StyleSet	ushort	16	
TargetWidgetID	ushort	16	
TargetParamID	ushort	16	
UnusedPad	N/A	16	0
FromValue	N/A	32	
ToValue	N/A	32	
AnimationLawRef	ushort	16	
Duration	ushort	16	
Delay	ushort	16	
AnimationRepetition	ushort	16	0 or 1: no repetition Maximum ushort value: repeat indefinitely Other values: specifies the number of repetitions

AnimationOnParam Event Structures:
A661_EVT_ANIMATION_STATUS_CHANGE are defined in Table 3.9.8-3.

Table 3.9.8-3 – AnimationOnParam Event Structures: A661_EVT_ANIMATION_STATUS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ANIMATION_STATUS_CHANGE
AnimationState	uchar	8	A661_COMPLETED A661_ABORTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.8-4.

Table 3.9.8-4 – AnimationOnParam Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
StyleSet	ushort	16	A661_STYLE_SET
FromValue	N/A	32	A661_FROM_VALUE
ToValue	N/A	32	A661_TO_VALUE
Duration	ushort	16	A661_ANIMATION_DURATION
Delay	ushort	16	A661_ANIMATION_DELAY
AnimationState	uchar	8	A661_ANIMATION_STATE
AnimationRepetition	ushort	16	A661_ANIMATION_REPETITION
AnimationLawRef	ushort	16	A661_ANIMATION_LAW_REF

3.9.9 AnimationRotation

Categories:

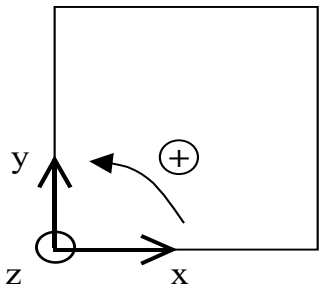
- Utility

Description:
The objective of this widget is to provide a means for performing a rotation animation on one widget.

The widget referenced in this AnimationRotation widget must be defined in the Definition File before the AnimationRotation widget.

The behavior for this widget is specified in Section 3.2.11.

Axis of rotation:

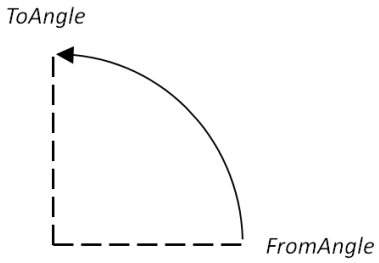


Angles are measured in degrees. Rotation is around the Z-axis. The ZERO degree is along the X-axis in the positive direction. The positive direction of angles is in the counter-clockwise direction from the X-axis.

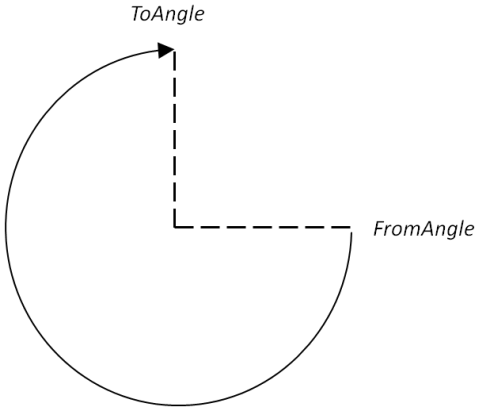
Direction and Number of turns:

The animation from FromAngle to ToAngle is performed clockwise or counter-clockwise, as specified by the Direction parameter. The NumberOfTurns parameter specifies how many times the FromAngle threshold is crossed. For example, with FromAngle = 0 and ToAngle = 90:

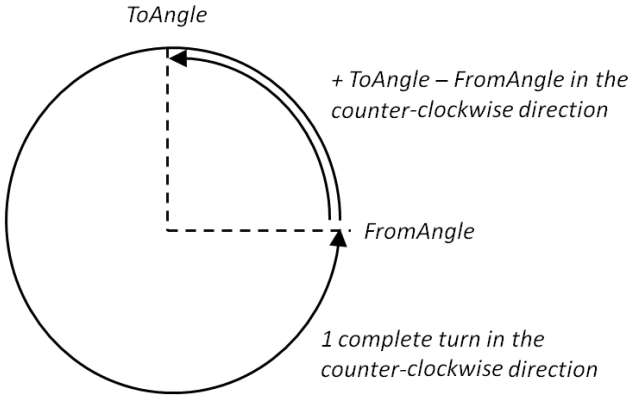
- If Direction = A661_COUNTER_CLOCKWISE, and NumberOfTurns = 0, then the Animation will perform from 0 to 90 as in the following diagram:



- If Direction = A661_CLOCKWISE, and NumberOfTurns = 0, then the Animation will perform from 0 to 90 as in the following diagram:



- If Direction = A661_COUNTER_CLOCKWISE, and NumberOfTurns = 1, then the Animation will start at FromAngle, do a complete turn in the counter-clockwise direction, and then perform from 0 to 90 as in the following diagram:



Notes:

1. **The rotation values do not change the ARINC 661 target widget parameter values. The result of the rotation is added to the coordinate system transformation specified by the widget position and rotation.**
2. **If this animation receives a A661_DEACTIVATE, it will put back the state of the rotation transformation to the identity transform (0 angle rotation).**

AnimationRotation Parameters are defined in Table 3.9.9-1.

Table 3.9.9-1 – AnimationRotation Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ANIMATION_ROTATION
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container 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		
TargetWidgetID	D	Widget ID which should be rotated according to this animation
CenterX	DR	X position of the center of the rotation **(relative to the target widget)**
CenterY	DR	Y position of the center of the rotation **(relative to the target widget)**
FromAngle	DR	The initial Angle of the rotation animation
ToAngle	DR	The final Angle of the rotation animation
AnimationLawRef	DR	The animation law reference to use for the animation (see Section 9).
Duration	DR	The duration of the animation in milliseconds
Delay	DR	The delay in milliseconds for starting the animation
Direction	DR	Direction of the animation
NumberOfTurns	DR	Number of turns of the animation
AnimationState	R	The state of the animation A661_PLAY A661_ABORT A661_END **A661_DEACTIVATE** **A661_FROM** **A661_TO**
AnimationRepetition	DR	Specifies how much times the animation will repeat (0 or 1 will mean no repetition, and the maximum value for ushort an indefinite repetition of the animation)

AnimationRotation Creation Structure is defined in Table 3.9.9-2.

Table 3.9.9-2 – AnimationRotation Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ANIMATION_ROTATION
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
TargetWidgetID	ushort	16	
StyleSet	ushort	16	
CenterX	long	32	
CenterY	long	32	
FromAngle	fr(180)	32	
ToAngle	fr(180)	32	
AnimationLawRef	ushort	16	
Duration	ushort	16	
Delay	ushort	16	
Direction	uchar	8	A661_CLOCKWISE A661_COUNTER_CLOCKWISE
NumberOfTurns	uchar	8	
AnimationRepetition	ushort	16	0 or 1: no repetition Maximum ushort value: repeat indefinitely Other values: specifies the number of repetitions
UnusedPad	**N/A**	**16**	**0**

AnimationRotation Event Structures: **A661_EVT_ANIMATION_STATUS_CHANGE** are defined in Table 3.9.9-3.

Table 3.9.9-3 – AnimationRotation Event Structures: A661_EVT_ANIMATION_STATUS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ANIMATION_STATUS_CHANGE
AnimationState	uchar	8	A661_COMPLETED A661_ABORTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.9-4.

Table 3.9.9-4 – AnimationRotation Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
FromAngle	fr(180)	32	A661_FROM_ANGLE
ToAngle	fr(180)	32	A661_TO_ANGLE
CenterX, CenterY	long x 2	32 x 2	A661_CENTER_XY
CenterX	long	32	A661_CENTER_X
CenterY	long	32	A661_CENTER_Y
StyleSet	ushort	16	A661_STYLE_SET
Duration	ushort	16	A661_ANIMATION_DURATION
Delay	ushort	16	A661_ANIMATION_DELAY
AnimationState	uchar	8	A661_ANIMATION_STATE
AnimationRepetition	ushort	16	A661_ANIMATION_REPETITION
AnimationLawRef	ushort	16	A661_ANIMATION_LAW_REF
Direction	uchar	8	A661_DIRECTION
NumberOfTurns	uchar	8	A661_NUMBER_OF_TURNS

3.9.10 AnimationScale

Categories:

- Utility

Description:
The objective of this widget is to provide a means for performing a scale animation on one widget.

The widget referenced in this AnimationScale widget must be defined in the Definition File before the AnimationScale widget.

The behavior for this widget is specified in Section 3.2.11.

Notes:

1. **The scale values do not change the ARINC 661 target widget parameter values. The result of the scale is added to the coordinate system transformation specified by the widget position and rotation.**
2. **If this animation receives a A661_DEACTIVATE, it will put back the state of the scaling transformation to the identity transform (1, 1 scale).**

AnimationScale Parameters are defined in Table 3.9.10-1.

Table 3.9.10-1 – AnimationScale Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ANIMATION_SCALE
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container 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		
TargetWidgetID	D	Widget ID which should be translated according to this animation

Parameters	Change	Description
FromScaleX	DR	The initial ScaleX of the scale animation
FromScaleY	DR	The initial ScaleY value of the scale animation
ToScaleX	DR	The final ScaleX of the scale animation
ToScaleY	DR	The final ScaleY value of the scale animation
AnimationLawRef	DR	The animation law reference to use for the animation (see Section 9)
Duration	DR	The duration of the animation in milliseconds
Delay	DR	The delay in milliseconds for starting the animation
AnimationState	R	The state of the animation A661_PLAY A661_ABORT A661_END **A661_DEACTIVATE** **A661_FROM** **A661_TO**
AnimationRepetition	DR	Specifies how much times the animation will repeat (0 or 1 will mean no repetition, and the maximum value for ushort an indefinite repetition of the animation)

AnimationScale Creation Structure is defined in Table 3.9.10-2.

Table 3.9.10-2 – AnimationScale Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ANIMATION_SCALE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
TargetWidgetID	ushort	16	
StyleSet	ushort	16	
FromScaleX	float	32	
FromScaleY	float	32	
ToScaleX	float	32	
ToScaleY	float	32	
AnimationLawRef	ushort	16	
Duration	ushort	16	
Delay	ushort	16	
AnimationRepetition	ushort	16	0 or 1: no repetition Maximum ushort value: repeat indefinitely Other values: specifies the number of repetitions

AnimationScale Event Structures: **A661_EVT_ANIMATION_STATUS_CHANGE** are defined in Table 3.9.10-3.

Table 3.9.10-3 – AnimationScale Event Structures: A661_EVT_ANIMATION_STATUS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ANIMATION_STATUS_CHANGE
AnimationState	uchar	8	A661_COMPLETED A661_ABORTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.10-4.

Table 3.9.10-4 – AnimationScale Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
FromScaleX	float	32	A661_FROM_SCALE_X
FromScaleY	float	32	A661_FROM_SCALE_Y
FromScaleX, FromScaleY	float x 2	32 x 2	A661_FROM_SCALE_XY
StyleSet	ushort	16	A661_STYLE_SET
ToScaleX	float	32	A661_TO_SCALE_X
ToScaleY	float	32	A661_TO_SCALE_Y
ToScaleX, ToScaleY	float x 2	32 x 2	A661_TO_SCALE_XY
Duration	ushort	16	A661_ANIMATION_DURATION
Delay	ushort	16	A661_ANIMATION_DELAY
AnimationState	uchar	8	A661_ANIMATION_STATE
AnimationRepetition	ushort	16	A661_ANIMATION_REPETITION
AnimationLawRef	ushort	16	A661_ANIMATION_LAW_REF

3.9.11 AnimationTranslation

Categories:

- Utility

Description:
The objective of this widget is to provide a means for performing a translation animation on one widget.

The widget referenced in this AnimationTranslation widget must be defined in the Definition File before the AnimationTranslation widget.

The behavior for this widget is specified in Section 3.2.11.

Notes:

- **The translation values do not change the ARINC 661 target widget parameter values position. The result of the translation is added to the coordinate system transformation specified by the widget position and rotation.** For example, if an AnimationTranslation is applied to a GpRectangle with a (PosX, PosY) position of (1000, 1000), and (ToValueX, ToValueY) = (4000, 4000), the graphical position of the GpRectangle at the end of the animation will be (5000, 5000)
- **If this animation receives a A661_DEACTIVATE, it will put back the state of the translation transformation to the identity transform (0, 0 translation).**

AnimationTranslation Parameters are defined in Table 3.9.11-1.

Table 3.9.11-1 – AnimationTranslation Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ANIMATION_TRANSLATION
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container 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		
TargetWidgetID	D	Widget ID which should be translated according to this animation
FromValueX	DR	The initial TranslationX of the translation animation
FromValueY	DR	The initial TranslationY value of the translation animation
ToValueX	DR	The final TranslationX of the translation animation
ToValueY	DR	The final TranslationY value of the translation animation
AnimationLawRef	DR	The animation law reference to use for the animation (see Section 9)
Duration	DR	The duration of the animation in milliseconds
Delay	DR	The delay in milliseconds for starting the animation
AnimationState	R	The state of the animation A661_PLAY A661_ABORT A661_END **A661_DEACTIVATE** **A661_FROM** **A661_TO**
AnimationRepetition	DR	Specifies how much times the animation will repeat (0 or 1 will mean no repetition, and the maximum value for ushort an indefinite repetition of the animation)

AnimationTranslation Creation Structure is defined in Table 3.9.11-2.

Table 3.9.11-2 – AnimationTranslation Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ANIMATION_TRANSLATION
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
TargetWidgetID	ushort	16	
StyleSet	ushort	16	
FromValueX	long	32	
FromValueY	long	32	
ToValueX	long	32	
ToValueY	long	32	
AnimationLawRef	ushort	16	
Duration	ushort	16	

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
Delay	ushort	16	
AnimationRepetition	ushort	16	0 or 1: no repetition Maximum ushort value: repeat indefinitely Other values: specifies the number of repetitions

AnimationTranslation Event Structures:
A661_EVT_ANIMATION_STATUS_CHANGE are defined in Table 3.9.11-3.

Table 3.9.11-3 – AnimationTranslation Event Structures: A661_EVT_ANIMATION_STATUS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ANIMATION_STATUS_CHANGE
AnimationState	uchar	8	A661_COMPLETED A661_ABORTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.11.4.

Table 3.9.11-4 – AnimationTranslation Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
FromValueX	long	32	A661_FROM_VALUE_X
FromValueY	long	32	A661_FROM_VALUE_Y
FromValueX, FromValueY	long x 2	32 x 2	A661_FROM_VALUE_XY
StyleSet	ushort	16	A661_STYLE_SET
ToValueX	long	32	A661_TO_VALUE_X
ToValueY	long	32	A661_TO_VALUE_Y
ToValueX, ToValueY	long x 2	32 x 2	A661_TO_VALUE_XY
Duration	ushort	16	A661_ANIMATION_DURATION
Delay	ushort	16	A661_ANIMATION_DELAY
AnimationState	uchar	8	A661_ANIMATION_STATE
AnimationRepetition	ushort	16	A661_ANIMATION_REPETITION
AnimationLawRef	ushort	16	A661_ANIMATION_LAW_REF

3.9.12 AnimationGroup

Categories:

- Utility
- Container

Description:
The objective of this widget is to provide a means for grouping several animations in parallel or sequentially. In either case, the duration of each child animation will be used.

The state transition algorithm for the widget is the following:

- If the AnimationState parameter is set to A661_PLAY:
  - **The children animations effects (the transformation or target parameter, if applicable) are initialized to the "FromValue".**
  - **Then the specified "Delay" is elapsed,**

  - **Then the AnimationGroup is started: The children animations are started in parallel or sequentially depending on the type of the AnimationGroup (this will apply the effect of the A661_PLAY for each child animation, except the initialization of the child animation effect to the "FromValue"),**
  - **Note that** if the animation is already playing, **nothing happens**.
- If the AnimationState parameter is set to A661_ABORT, the animation is aborted immediately if it is already playing. **The A661_ABORT is also propagated to the children of the animation.**
- If the AnimationState parameter is set to A661_END, the animation is **ended** immediately if it is already playing. **The A661_END is also propagated to the children of the animation.**
- **If the AnimationState parameter is set to A661_DEACTIVATE, the animation is deactivated immediately. The A661_DEACTIVATE is also propagated to the children of the animation.**
- **If the AnimationState parameter is set to A661_FROM and the animation is not currently playing, the A661_FROM is propagated to the children of the animation.**
- **If the AnimationState parameter is set to A661_TO and the animation is not currently playing, the A661_TO is propagated to the children of the animation.**
- If the Animation is playing and all the children animations have ended and the AnimationRepetition parameter has the 0 or 1 value, then the CDS send the event A661_EVT_ANIMATION_STATUS_CHANGE with the value A661_COMPLETED.
- If the Animation is playing and all the children animations have ended and the AnimationRepetition parameter value is the maximum value for ushort, then the Animation is starting again. The CDS will never send the event A661_EVT_ANIMATION_STATUS_CHANGE.
- If the Animation is playing and all the children animations have ended and the AnimationRepetition parameter value is different from 0, 1, or the maximum value for ushort, the Animation is starting again until all the repetitions has been performed. At the end of the last repetition, the CDS send the event A661_EVT_ANIMATION_STATUS_CHANGE with the value A661_COMPLETED.
- **If a child Animation of a sequential AnimationGroup is aborted, deactivated or ended, then the Animation goes to the next child Animation of the AnimationGroup.**

AnimationGroup type:

- **If the type of the AnimationGroup is A661_PARALLEL, all the children Animation are started in parallel**.
- If the type of the AnimationGroup is A661_SEQUENTIAL, the children Animation are started sequentially in the order of the Definition File. Note that the precise behavior of an AnimationGroup of this type is implementation dependent.

- **If the type of the AnimationGroup is A661_SEQUENTIAL_FROM, the children Animation widgets are recursively set to their "FromValue" right before being started sequentially in the order of the Definition File.**

  Notes:

  1. Each child animation will still be started after its own delay and its own duration, additionally with the delay for the group. For example, if the AnimationGroup has a Delay of 500 ms, and one child animation has a delay of 200 ms, then this child animation will start after a delay of 700 ms = 500 ms + 200 ms after the start of the AnimationGroup
  2. Changing the parameters of the animation during the playing of this animation has no effect for the current animation

AnimationGroup Parameters are defined in Table 3.9.12-1.

Table 3.9.12-1 – AnimationGroup Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_ANIMATION_GROUP
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Anonymous	D	Ability to be modified at run-time by the UA
Specific parameters		
Delay	DR	The delay in milliseconds for starting the animation
Type	D	Indicates if the children animations are applied in parallel or sequentially A661_PARALLEL A661_SEQUENTIAL **A661_SEQUENTIAL_FROM**
AnimationState	R	The state of the animation A661_PLAY A661_ABORT A661_END **A661_DEACTIVATE** **A661_FROM** **A661_TO**
AnimationRepetition	DR	Specifies how much times the animation will repeat (0 or 1 will mean no repetition, and the maximum value for ushort an indefinite repetition of the animation)

AnimationGroup Creation Structure is defined in Table 3.9.12-2.

Table 3.9.12-2 – AnimationGroup Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_ANIMATION_GROUP
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Type	uchar	8	
Delay	ushort	16	
AnimationRepetition	ushort	16	0 or 1: no repetition Maximum ushort value: repeat indefinitely Other values: specifies the number of repetitions

AnimationGroup Event Structures: **A661_EVT_ANIMATION_STATUS_CHANGE** are defined in Table 3.9.12-3.

Table 3.9.12-3 – AnimationGroup Event Structures: A661_EVT_ANIMATION_STATUS_CHANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ANIMATION_STATUS_CHANGE
AnimationState	uchar	8	A661_COMPLETED A661_ABORTED
UnusedPad	N/A	8	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.12-4.

Table 3.9.12-4 – AnimationGroup Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Delay	ushort	16	A661_ANIMATION_DELAY
AnimationState	uchar	8	A661_ANIMATION_STATE
AnimationRepetition	ushort	16	A661_ANIMATION_REPETITION

3.9.13 MapVert_Container

Categories:

- Container
- Map management

Description:
The MapVert_Container is used to position a set of non-interactive widgets on a map using real world coordinates (e.g., Distance/Altitude). The X and Y parameters provide this real world position data. Their interpretation is dependent on the MapDataFormatX and MapDataFormatY parameters of the parent MapVert_Source widget. The screen location resulting from the projection of the X and Y parameters defines the origin of the MapVert_Container's children. The position and size of the children are still defined in hundredth of millimeters.

The MapVert_Container scales its children according to the widgets RangeX/Y and SizeX/Y parameters as well as the RangeX/Y and SizeX/Y parameters of the ancestor MapVert.

Behavior of this widget before the X, Y, RangeX, RangeY, SizeX, and SizeY parameters are received is implementation dependent.

The MapVert_Container sends events to its owner to indicate when its children widgets are onscreen or offscreen. This is calculated based on the RangeX/Y parameter.

Restriction:
A MapVert_Container must be in a MapVert_Source container.

MapVert_Container Parameters are defined in Table 3.9.13-1.

Table 3.9.13-1 – MapVert_Container Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAPVERT_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of the widget
SizeX	R	Distance in screen units (hundredth of mm) equivalent to RangeX
SizeY	R	Distance in screen units (hundredth of mm) equivalent to RangeY
Specific parameters		
X	R	Dependent on MapDataFormat. See Table 3.4.4-2b
Y	R	Dependent on MapDataFormat. See Table 3.4.4-2b
MapSynchronizationNumber	R	See Section 3.2.8.4
RangeX	R	Geo-referenced range in nm
RangeY	R	Geo-referenced range in feet

MapVert_Container Creation Structures are defined in Table 3.9.13-2.

Table 3.9.13-2 – MapVert_Container Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPVERT_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
UnusedPad	N/A	8	0
Visible	uchar	8	A661_FALSE A661_TRUE

MapVert_Container Event Structures: A661_EVT_ONSCREEN is defined in Table 3.9.13-3.

Table 3.9.13-3 – MapVert_Container Event Structures: A661_EVT_ONSCREEN

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ONSCREEN
UnusedPad	N/A	16	0

MapVert_Container Event Structures: A661_EVT_OFFSCREEN is defined in Table 3.9.13-4.

Table 3.9.13-4 – MapVert_Container Event Structures: A661_EVT_OFFSCREEN

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_OFFSCREEN
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.13-5.

Table 3.9.13-5 – MapVert_Container Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
X	Dependent on MapDataFormat	32	A661_REF_POS_X
Y	Dependent on MapDataFormat	32	A661_REF_POS_Y
X, Y	Dependent on MapDataFormat	32 x 2	A661_REF_POS_XY
RangeX	fr(32768)	32	A661_RANGE_X
RangeY	long	32	A661_RANGE_Y
RangeX, RangeY	fr(32768) long	32 x 2	A661_RANGE_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER See Section 3.2.8.4

3.9.14 MapVert_Panel

Categories:

- Container
- Map management
- Graphical Representation

This container can be embedded into a **MapVert_Source** widget. The X and Y parameters provide this real world position data. Their interpretation is dependent on the MapDataFormatX and MapDataFormatY parameters of the parent MapVert_Source widget. The screen location resulting from the projection of the X and Y parameters defines the origin of the MapVert_Panel's children.

Its aim is to provide geographic-aware interactivity, e.g., it is possible to add a few buttons close to a waypoint or a text entry close to an airport.

The unit for the widgets within the container remains the standard hundredth of mm A661 unit. The size of the widget can be used by the CDS to manage specific placement policies.

Table 3.9.14-1 – MapVert_Panel Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661 MAPVERT_PANEL
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Enable	DR	Ability of the widget's descendants to be interactive.
Visible	DR	Visibility of the widget.
SizeX	DR	The X dimension size (width) of the widget in hundredths of mm.
SizeY	DR	The Y dimension size (height) of the widget in hundredths of mm.
StyleSet	DR	Reference to predefined graphical characteristics inside CDS.
Specific parameters		
X	DR	Dependent on MapDataFormat See Table 3.4.4-2b.
Y	DR	Dependent on MapDataFormat See Table 3.4.4-2b.
MapSynchronizationNumber	R	See Section 3.2.8.4.

MapVert_Panel Creation Structure is defined in Table 3.9.14-2.

Table 3.9.14-2 – MapVert_Panel Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAPVERT_PANEL
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
X	Dependent on MapDataFormat	32	See Table 3.4.4-2b
Y	Dependent on MapDataFormat	32	See Table 3.4.4-2b
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
UnusedPad	N/A	16	0

The MapVert_Panel widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.9.14-3.

Table 3.9.14-3 – A661_MapVert_Panel Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
X, Y	long x 2	32 x 2	A661_REF_POS_XY
X	long	32	A661_REF_POS_X
Y	long	32	A661_REF_POS_Y
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
MapSynchronizationNumber	ushort	16	A661_MAP_SYNCHRONIZATION_NUMBER see Section 3.2.8.4

3.9.15 MapBoundary

Categories:

- Container
- Map management

Description:
The MapBoundary widget defines a region of a map widget. Typically, this could be used to draw certain map items along the edge of the region when they would otherwise be drawn outside of the region. The exact impact of the MapBoundary on map items is implementation dependent.

The region is defined by a series of sections. Each section is made up of a BeginAngle, Distance, and Algorithm parameter. Each section spans from its BeginAngle to the BeginAngle of the next section. The center of the region is defined by the ScreenReferencePoint of the parent MapHorz widget. Available values for Algorithm are Arc, Vert_Line, and Horz_Line. The distance between the section and the center of the region is defined by the Distance parameter. The meaning of that distance is dependent on the Algorithm parameter (i.e., radius for Arc, perpendicular distance for Vert_Line, and Horz_Line. The region is assumed to be defined in the counterclockwise direction and closed (i.e., the last section spans from its BeginAngle to the BeginAngle of the first section). Although the definition is guaranteed to begin and end at the same angle, a gap could still be present in case the distance is different between **adjacent sections**. In this case, the behavior related to the gap is implementation dependent. **For example, a Parkable_Symbol could intuitively be parked at either end of the gap, but Parking_Line_Start/Parking_Line_End may not have an intuitive parking location if the parking line crosses a gap in the boundary.** A reverse in the direction of BeginAngle between sections implies a concave section. The MapBoundary can be effectively disabled by setting the NumberOfEntries parameter to zero. Returning to the previous value restores the MapBoundary effectiveness.

The map items impacted by the MapBoundary widget are:

PARKABLE_SYMBOL

PARKABLE_SYMBOL_INTERACTIVE

PARKABLE_SYMBOL_ROTATED

PARKABLE_SYMBOL_ROTATED_INTERACTIVE

PARKABLE_SYMBOL_TARGET

PARKABLE_SYMBOL_TARGET_INTERACTIVE

PARKING_LINE_START

PARKING_LINE_END

Figure 3.9.15-1 shows the ray between the screen reference point and the symbol coordinate intersects the map boundary edges in three places. The drawn location of the symbol could be one of these locations depending on implementation dependent behavior.

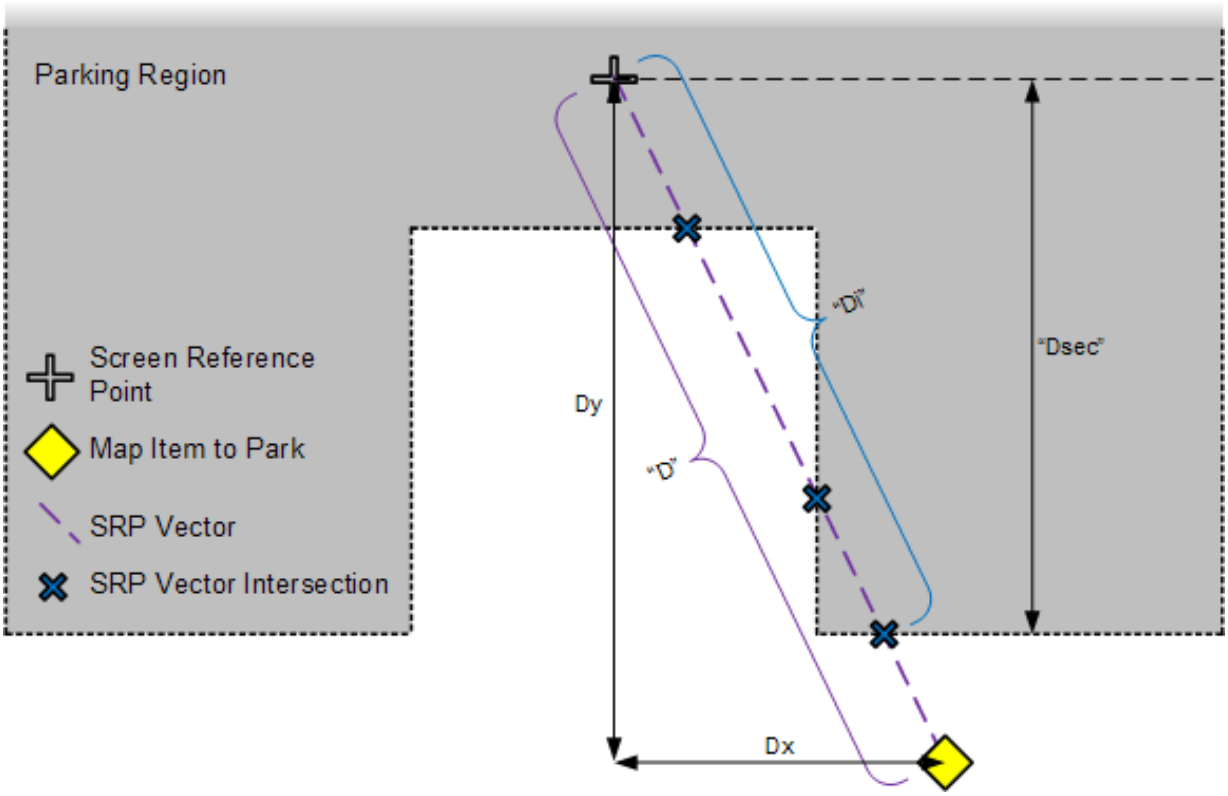


Figure 3.9.15-1 – Parkable Symbol Example

Figure 3.9.15-2 shows three nominal parking line cases, one with both ends parked, one with both ends unparked, and a third with one end parked and the other unparked.

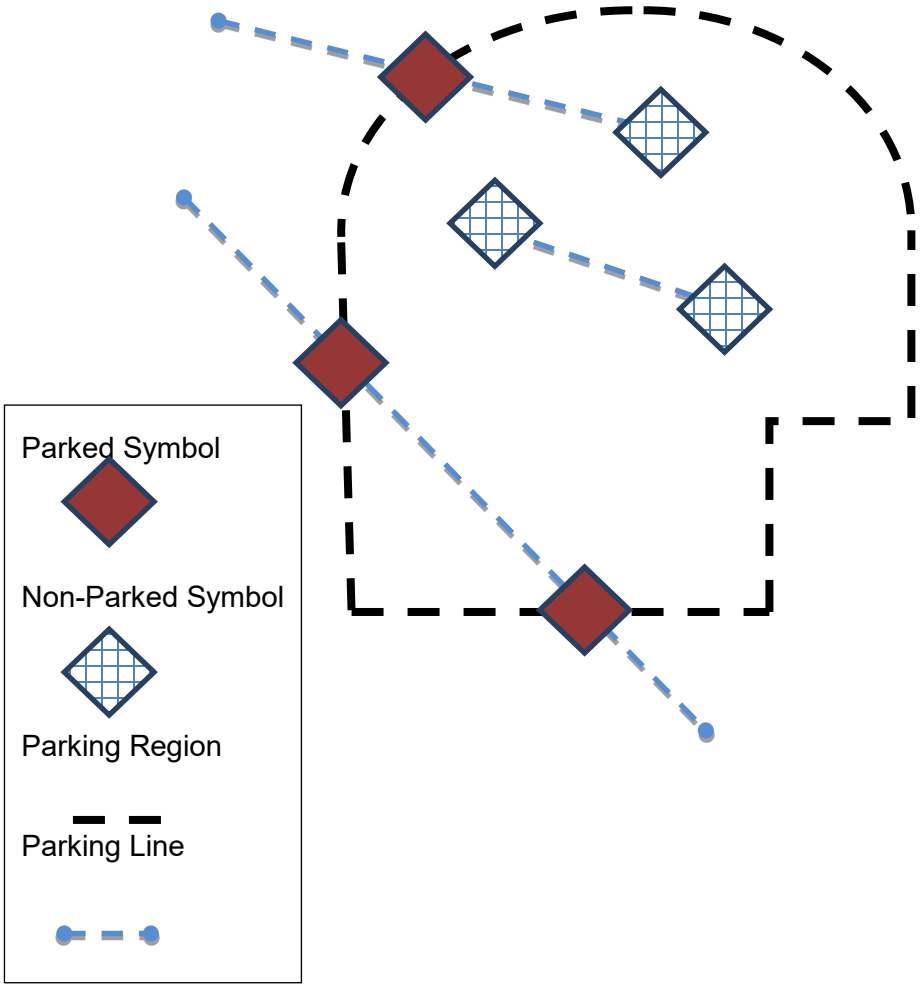


Figure 3.9.15-2 – Parking Line Example

Figure 3.9.15-3 shows a parking line that is within a boundary that has a gap. The gap is present because the distance parameter of the last section is different that the distance parameter of the first section. This is not an expected way to define a boundary and the behavior in this case is implementation dependent.

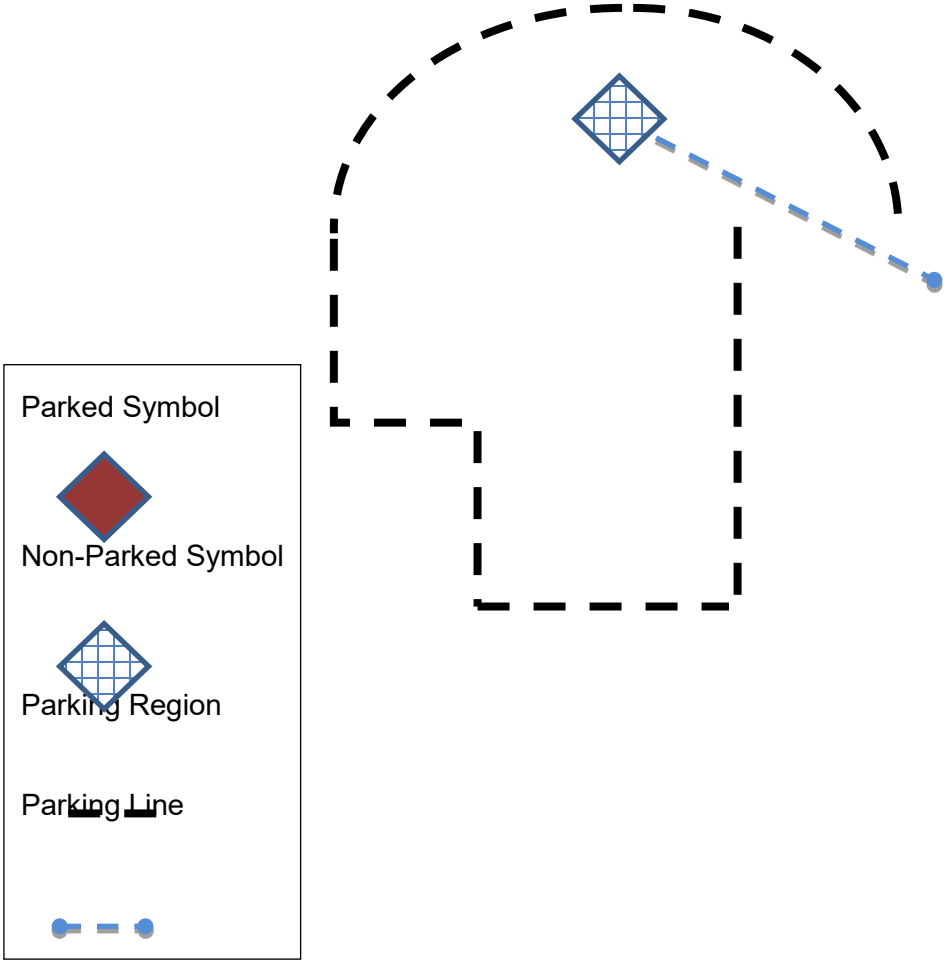


Figure 3.9.15-3 – Parking Line Example With Gap

Restriction:
A MapBoundary widget must be a child of a MapHorz widget.

MapBoundary Parameters are defined in Table 3.9.15-1.

Table 3.9.15-1 – MapBoundary Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAP_BOUNDARY
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 interactive
Specific Parameters		
MaxNumberOfEntries	D	Maximum number of sections
NumberOfEntries	DR	Number of sections currently activated (set to zero to disable the effectiveness of the boundary)
BeginAngleArray [NumberOfEntries]	D	Starting angle **(relative to screen)** of each section relative to the last one in a counter-clockwise direction
DistanceArray [NumberOfEntries]	D	Distance **(1/100 mm)** of the section from the reference point
AlgorithmArray [NumberOfEntries]	D	Array of algorithm for each section
BoundaryArray	R	An array of angle/distance/algorithm triplets to change the boundary at runtime

MapBoundary Creation Structures are defined in Table 3.9.15-2.

Table 3.9.15-2 – MapBoundary Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAP_BOUNDARY
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
BeginAngleArray [NumberOfEntries]	{fr(180)}+	32 * NumberOf Entries	
DistanceArray [NumberOfEntries]	{ulong}+	32 * NumberOf Entries	
AlgorithmArray [NumberOfEntries]	{ulong}+	32 * NumberOf Entries	A661_ARC A661_HORZ_LINE A661_VERT_LINE

The MapBoundary widget sends no events.

Available SetParameter identifiers and associated data structure are defined in Table 3.9.15-3.

Table 3.9.15-3 – MapBoundary Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Enable	uchar	8	A661_ENABLE
Visible	uchar	8	A661_VISIBLE
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
BoundaryArray	N/A	{32}+	A661_BOUNDARY_ARRAY

Table 3.9.15-4 – A661_ParameterStructure BoundaryArray

A661_PARKING_ARRAY_Structure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_BOUNDARY_ARRAY
NumberOfEntriesUpdated	ushort	16	
{BoundaryList_Structure}+			Refer to BoundaryList_Structure.

Table 3.9.15-5 – BoundaryList_Structure

BoundaryList_Structure	Type	Size (bits)	Description
SectionIndex	ushort	16	Index of the section
UnusedPad	N/A	16	
BeginAngle	fr(180)	32	Begin angle of the section
Distance	ulong	32	Sections' distance from the map center
Algorithm	ulong	32	Algorithm used to calculate maximum placement distance.

3.9.16 DataConnector

This section relocated from the former Appendix H, Section H-6.2 and enhanced.

Categories:

- Utility

Description:
The DataConnector is similar to a regular Connector widget, but in addition, it provides the ability for an array of parametric information to be passed to the Application that owns the referenced layer.

This type of information can be used in situations when a Layer can be drawn in one of several formats (e.g., half-screen versus full-screen). This type of information can also be used in situations where some information about the context of the parent layer is necessary for a user application to draw its layer (e.g., the current PRP and Range of the MapHorz widget which is the parent of the DataConnector widget).

The parametric data that is sent does not need to be interpreted by the CDS. The meaning of the parametric data only needs to be known by the designer creating the layer in which the DataConnector resides and the designer of the layer to which the connector points.

The CDS sends the A661_NOTE_CONNECTOR_DATA notification in the following situations:

- Following A661_NOTE_LAYER_ACTIVE
- When the Data parameter is changed (and the layer is active)
- Implementation dependent situations, e.g., cyclically

This notification contains the block of parametric data.

Restriction:

- None

DataConnector Parameters are defined in Table 3.9.16-1.

Table 3.9.16-1 – DataConnector Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_DATA_CONNECTOR
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's descendants to be interactive
Specific parameters		
Connector Reference	D	Identifier of or Reference to the Connected layer. The resolution of the link between the DataConnector and the layer is implementation dependent. All events generated by the widgets of the connected (child) layer are still handled by the owning application of that layer.
DataLength	D	Number of 32 bit words in the Data parameter
Data	DR	Data to be passed to the referenced layer.

DataConnector Creation Structure is defined in Table 3.9.16-2.

Table 3.9.16-2 – DataConnector Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_DATA_CONNECTOR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
ConnectorReference	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
DataLength	ushort	16	
Data	N/A	32 * DataLength	

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

Available SetParameter identifiers and associated data structure are defined in Table 3.9.16-3.

Table 3.9.16-3 – DataConnector Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
Data	N/A	{32}+	A661_APP_DATA

3.9.17 EventHandler

Categories:

- Utility

Description:
The EventHandler widget defines how CDS should handle an event without the UA needing to send Set Parameter commands after receiving the event. This widget forms a logical connection between an event and widget parameters for cases in which a UA responds to an event the same way each time.

This widget offers two main benefits:

- Reduced bandwidth utilization by reducing the number of Set Parameter commands needing to be sent or reducing the size of BufferFormat parameter lists.
- Reduced latency since the UA does not need to receive an event and subsequently set widget parameters accordingly to update the display; CDS can process the event internally based on the instructions provided by the UA at definition time.

In order for interactive widget states to remain synchronized between the UA and CDS, the event that prompts the CDS to execute the EventHandler is still sent to the UA, as well as any other events that are generated by applying the EventHandler. For example, consider a use case in which a list of buttons is presented with “Check all” and “Clear all” buttons. If an EventHandler sets the CheckButtonState for each button based on “Check all” being selected, the “Check all” button selection is sent to the UA as well as any resulting selection event for each button widget whose state was set as a result of the EventHandler. In this way, the EventHandler is similar to the ProxyButton widget, which sends events to multiple widgets when more than one ProxyButton is assigned to a single key.

For widget parameters not reflecting an interaction state, it is assumed that the EventHandler widget will be used in cases when the UA does not manage or track the value of the effected widget parameters (such as Visible). Typically, widget parameters identified in an EventHandler should not be set separately by the UA.

The figure below presents a common use case. The selection of a CheckButton widget results in the subsequent display of other widgets, in this case 4 Label widgets. Without an EventHandler widget, the UA must receive an event that the CheckButton was selected, and then set the visibility of the Label widgets after the event is processed. With an EventHandler widget, the Visible parameters of the Label widgets are set by CDS when the CheckButtonState of the CheckButton widget is changed.

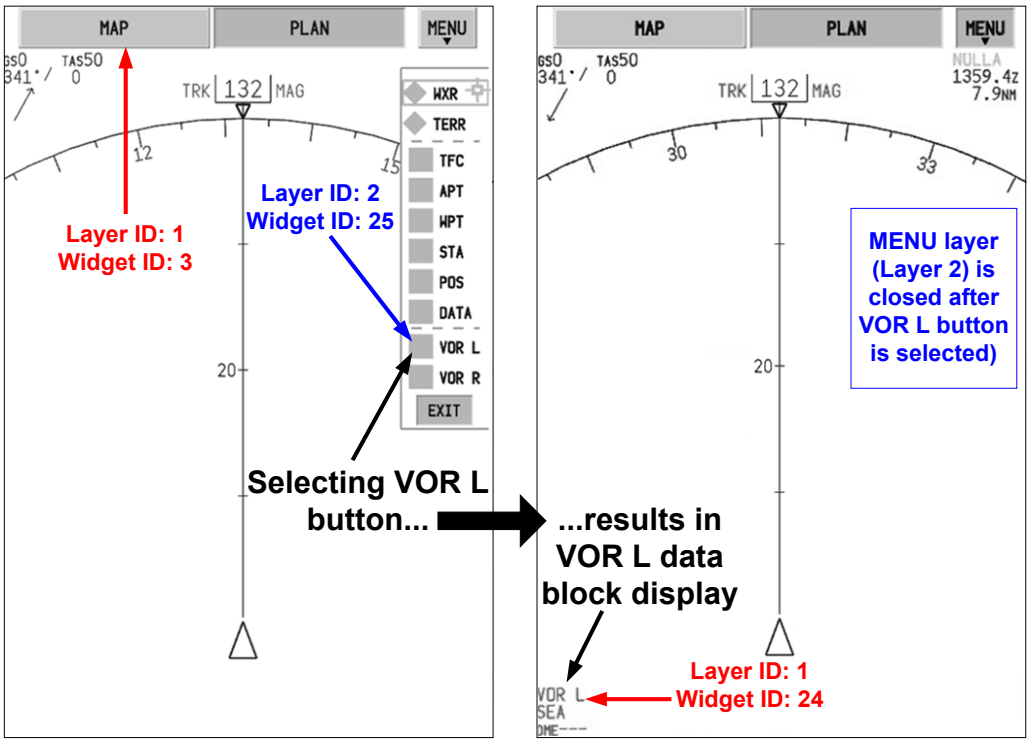


Figure 3.9.17-1 – EventHandler Example

The TriggerLayerIdent, TriggerWidgetIdent, and TriggerEvent parameters (collectively referred to as the “trigger”) indicate the widget event that triggers the EventHandler widget to perform the defined actions. In the example above, the A661_EVT_STATE_CHANGE event generated from the “VOR L” CheckButton widget is the trigger.

Two types of resulting actions can be defined: widget parameter results (defined by WidgetParamResultsArray) and layer request results (defined by LayerReqResultArray). The number of entries in each array is defined by NumWidgetParamResults and NumLayerReqResults, respectively. If one of these parameters is unused (e.g., no layer requests result from the event), the Num parameter is set to 0 and the associated ResultArray is omitted.

The WidgetParamResultArray parameter can be used to define individual widget parameters to be set by the EventHandler. For each entry, a destination widget parameter is referenced with the DestLayerIdent, DestWidgetIdent, and DestParamIdent fields. In the example above, the “VOR L” Label widget’s Visible parameter is the destination.

The logic to use for setting the Destination parameter can be either one of the standard defined operators, or an implementation dependent operation. The LogicOperatorCode defines the type of operation to perform, such as an Assignment or Sum. Implementation dependent logic operator code values can be defined starting with the value 0x80. The following table lists the operator codes defined.

Table 3.9.17-1 – Logic Operator Codes

Operator keyword	Value of DestParamIdent =			Comment
A661_OPERATOR_ASSIGNMENT	SourceParamIdent			
A661_OPERATOR_INVERSE	NOT(SourceParamIdent)			Not a bitwise operation, any non-zero value returns A661_FALSE and zero returns A661_TRUE
A661_OPERATOR_CONSTANT	Argument			
A661_OPERATOR_SUM	SourceParamIdent	+	Argument	
A661_OPERATOR_PRODUCT	SourceParamIdent	*	Argument	
A661_OPERATOR_LOGICAL_AND	SourceParamIdent	&&	Argument	Not a bitwise operation, any non-zero value is treated as True
A661_OPERATOR_LOGICAL_OR	SourceParamIdent	||	Argument	Not a bitwise operation, any non-zero value is treated as True
A661_OPERATOR_ADD_ASSIGN_SRC	DestParamIdent	+	SourceParamIdent	
A661_OPERATOR_SUB_ASSIGN_SRC	DestParamIdent	-	SourceParamIdent	
A661_OPERATOR_PROD_ASSIGN_SRC	DestParamIdent	*	SourceParamIdent	
A661_OPERATOR_ADD_ASSIGN_ARG	DestParamIdent	+	Argument	
A661_OPERATOR_SUB_ASSIGN_ARG	DestParamIdent	-	Argument	
A661_OPERATOR_PROD_ASSIGN_ARG	DestParamIdent	*	Argument	

Note: SourceParamIdent and DestParamIdent in the above table represent the current value for that parameter.

Depending on the LogicOperatorCode, the logic may include the use of a Source widget parameter and/or an Argument field. The Source widget parameter is references by the SourceLayerIdent, SourceWidgetIdent, and SourceParamIdent set of fields. In the above example, the “VOR L” CheckButton’s CheckButtonState is the Source to use in the logic. Note that the Destination parameter can also be used as the Source parameter (for example, for use with an Inverse operator).

The Argument parameter is a 32-bit field that is meant to be able to accommodate any numeric data type using up to 32 bits, since different operations may desire different data types, but its use should be consistent with the data types of the needs of the destination (and source, if applicable) parameter data types. If fewer than 32 bits are needed, the utilized bytes should populate the least significant bytes of the field (i.e., a uchar in the least signification byte, and a short/ushort in the two least significant bytes). CDS must mask out other bits (e.g., bitwise AND with 0x0000FFFF when used with a 16-bit value) prior to using the Argument in an operation with data types smaller than 32 bits. Negative values should be sign-extended.

The Source and Destination data types should be the same. The result of an operation in which the Source or Argument data type is not the same as the Destination data type is implementation dependent. UA designers should select operators and arguments that will not result in an overflow. However, if overflow occurs in an arithmetic operation, the result is implementation dependent.

In the example depicted above, the LogicOperatorCode of 0 (Assignment) is used. Therefore, the “VOR L” Label widget’s Visible parameter will be assigned the value of the “VOR L” CheckButton widget’s CheckButtonState parameter. The resulting behavior is that the Label widget is always displayed when the CheckButton is selected and removed when the CheckButton is unselected.

EventHandler operations are sequential atomic functions; therefore, any subsequent operations are based on previous results. For example, assume the Visible parameter of widget A was changed from False to True in the first

WidgetParamResultArray operation, and widget A's Visible parameter is defined as the Source in an operation for widget B's Enable parameter. The logic to set widget B's Enable parameter would then use a Source value of True instead of False.

Layer requests are handled before the set widget parameter commands.

The LayerReqResultArray parameter can be used to define layer requests to be generated by the EventHandler. For each entry, the Layer for which the request is being made is set by the LayerIdent field. Since not all layer request structures are the same size, the CommandSize field provides the size for that particular entry. The RequestStructure field is populated with the desired layer request structure, following the structure definitions in Section 4.5.3.3.

For example, assume that menu window containing the Check Buttons in the figure above is defined as its own layer, and is automatically closed after the CheckButton is pressed. A A661_REQ_LAYER_INACTIVE structure and reference to that menu's layer can be included in the LayerReqResultArray.

For any unused field in a particular array structure instance, the size of the field should be filled with zeroes, and it is assumed that the CDS will ignore these fields.

Since the EventHandler intentionally eliminates the need for the UA to be in the loop on the defined operations, it is not a good fit for events requiring UA validation. UA designers should not use any widget parameters requiring UA validation as the trigger.

Parameter value changes resulting from applying an EventHandler should be treated like a basic Set Parameter commands. Therefore, CDS should still apply any UA Set Parameter commands for a widget parameter included in an EventHandler. For example, if the Label widgets drawing the "DME" and "----" strings in the above figure have their Visible parameter set to True due to an EventHandler based on selection of the "VOR L" CheckButton, but then are set to False via regular Set Parameter commands, then their Visible states are False.

The EnableHandler parameter dictates whether the EventHandler is invoked.

EventHandler operations are processed in the order they appear in the Definition File. If an EventHandler widget contains invalid widget and/or parameter references, the result is implementation dependent.

COMMENTARY

If an event is generated based on setting a run-time modifiable parameter, using EventHandlers to set widget parameters that trigger events themselves could result in an undesired feedback loop. It is recommended to restrict the destination parameters in the WidgetParamResultArray to those that do not generate subsequent events.

Restrictions:

- None

Event Handler parameters are defined in Table 3.9.17-2.

Table 3.9.17-2 – EventHandler Parameters

Parameters	Change	Description
WidgetType	D	A661_EVENT_HANDLER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
EnableHandler	DR	If True, Event Handler is on. Otherwise, Event Handler is inhibited when TriggerEvent occurs on TriggerWidget.
TriggerLayerIdent	D	Layer containing the trigger widget
TriggerWidgetIdent	D	Identifier of the widget for which the TriggerEvent will trigger the CDS to set the parameters defined in BufferStructure to the values defined in BufferOfParameter
TriggerEvent	D	Type of event (using keyword defined in Table 4.6-9)
NumWidgetParamResults	D	Number of widget parameters set due to the event
NumLayerReqResults	D	Number of layer requests processed due to the event
WidgetParamResultArray	D	Defines widget parameters set by EventHandler, including logic operator with source widget parameter and argument to use.
LayerReqResultArray	D	Defines layer requests invoked by EventHandler. No logic is applied (if EventHandler is invoked, the UA requests are processed).

Event Handler Creation Structure is defined in Table 3.9.17-3.

Table 3.9.17-3 – EventHandler Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_EVENT_HANDLER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	0
EnableHandler	uchar	8	A661_FALSE A661_TRUE
TriggerLayerIdent	uchar	8	
TriggerWidgetIdent	ushort	16	
TriggerEvent	ushort	16	See Table 4.6-9
UnusedPad	ushort	16	0
NumWidgetParamResults	uchar	8	
NumLayerReqResults	uchar	8	
WidgetParamResultArray [NumWidgetParamResults]	Array of WidgetParamResult_Struct	{32}+	
LayerReqResultArray [NumLayerReqResults]	Array of LayerReqResult_Struct	{32}+	

The WidgetParamResult_Struct structure is defined in Table 3.9.17-4.

Table 3.9.17-4 – WidgetParamResult Structure

Field	Type	Size (bits)	Value/Range When Necessary
LogicOperatorCode	uchar	8	See Table 3.9.8-1 for list of standard operators. The set of LogicOperatorCode values may be extended. The range of 0x00-0x7F is reserved for this specification. OEM customization operator code values may be defined using the range 0x80 through 0xFF.
DestLayerIdent	uchar	8	Layer containing the widget whose parameter is set by Event Handler
DestWidgetIdent	ushort	16	Widget whose parameter is to be set by Event Handler
DestParamIdent	ushort	16	Parameter whose value is set by Event Handler. See Table 4.6-8
UnusedPad	N/A	16	0
Argument	int	32	Constant value to be used with logic
Source	Dependent on SourceType	{32}+	Use SourceWidgetParam_Struct if SourceType = A661_SOURCE_WIDGET_PARAM

The SourceWidgetParam_Struct structure is defined in Table 3.9.17-5.

Table 3.9.17-5 – SourceWidgetParam_Struct Creation Structure

Field	Type	Size (bits)	Value/Range When Necessary
SourceType	uchar	8	A661_SOURCE_WIDGET_PARAM: Source is defined by a structure including SouceLayerIdent, SourceWidgetIdent, and SourceParamIdent. No other types supported at this time. The set of SourceType values may be extended. The range of 0x00-0x7F is reserved for this specification. OEM customization SourceType values may be defined using the range 0x80 through 0xFF.
SourceLayerIdent	uchar	8	Layer containing the widget whose parameter is to be used in logic.
UnusedPad	N/A	16	0
SourceWidgetIdent	ushort	16	Widget whose parameter is to be used in logic.
SourceParamIdent	ushort	16	Parameter whose value is used in logic. See Table 4.6-8.

The LayerReqResult_Struct structure is defined in Table 3.9.17-6.

Table 3.9.17-6 – LayerReqResult Creation Structure

Field	Type	Size (bits)	Value/Range When Necessary
LayerIdent	uchar	8	ID of layer to which request applies
UnusedPad	N/A	8	0
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.

Event Handler run-time parameters are defined in Table 3.9.17-7.

Table 3.9.17-7 – EventHandler Run-Time Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
EnableHandler	uchar	8	A661_ENABLE_HANDLER

The EventHandler widget does not send any events of its own.

Table 3.9.17-8 provides an example definition of an EventHandler widget instance corresponding to Figure 3.9.17-1.

Table 3.9.17-8 – EventHandler Example

CreateParameterBuffer field	Value	Notes
WidgetType	0xA600	A661_EVENT_HANDLER
WidgetIdent	100	Ident of EventHandler widget
ParentIdent	0	EventHandler parent is always a layer
EnableHandler	1	A661_TRUE
TriggerLayerIdent	2	Layer with “VOR L” CheckButton
TriggerWidgetIdent	25	“VOR L” CheckButton
TriggerEvent	0xE070	A661_EVT_STATE_CHANGE
UnusedPad	0	
NumWidgetParamResults	1	1 WidgetParamResultArray entry
NumLayerReqResults	1	1 LayerReqResultArray entry
WidgetParamResultArray [NumWidgetParamResults]	See below	
LayerReqResultArray [NumLayerReqResults]	See below	

WidgetParamResultArray field	Value	Notes
LogicOperatorCode	0	A661_OPERATOR_ASSIGNMENT
DestLayerIdent	1	Layer with “VOR L” Label
DestWidgetIdent	24	Ident of “VOR L” Label
DestParamIdent	0xB530	A661_VISIBLE
UnusedPad	0	
Argument	0	Not used with LogicOperatorCode = 0
Source	See below	

Source field	Value	Notes
SourceType	0	A661_SOURCE_WIDGET_PARAM
SourceLayerIdent	2	Layer with “VOR L” CheckButton
UnusedPad	0	
SourceWidgetIdent	25	Ident of “VOR L” CheckButton
SourceParamIdent	0xB244	A661_INNER_STATE_CHECK

LayerReqResultArray field	Value	Notes
LayerIdent	2	Layer to which request applies
UnusedPad	0	
CommandSize	8	Size of A661_REQ_LAYER_INACTIVE
RequestStructure	See below	
RequestStructure field	Value	Notes
A661_REQ_LAYER_INACTIVE	0xDA02	Make Layer 2 inactive
UnusedPad	0	

3.9.18 FramingRefreshContainer

Categories:

- Container
- Utility

Description:
The FramingRefreshContainer has no graphical representation. Its purpose is to ensure that its children are rendered together, even though updates of its children are sent in separate set parameter commands.

This widget has three states:

1. Direct Update – All Set Parameter Commands are instantly consumed
2. Caching – The effect of Set Parameter Commands is remembered/calculated but not rendered
3. Frame Update – Cached changes are rendered, before moving back to the Caching state.

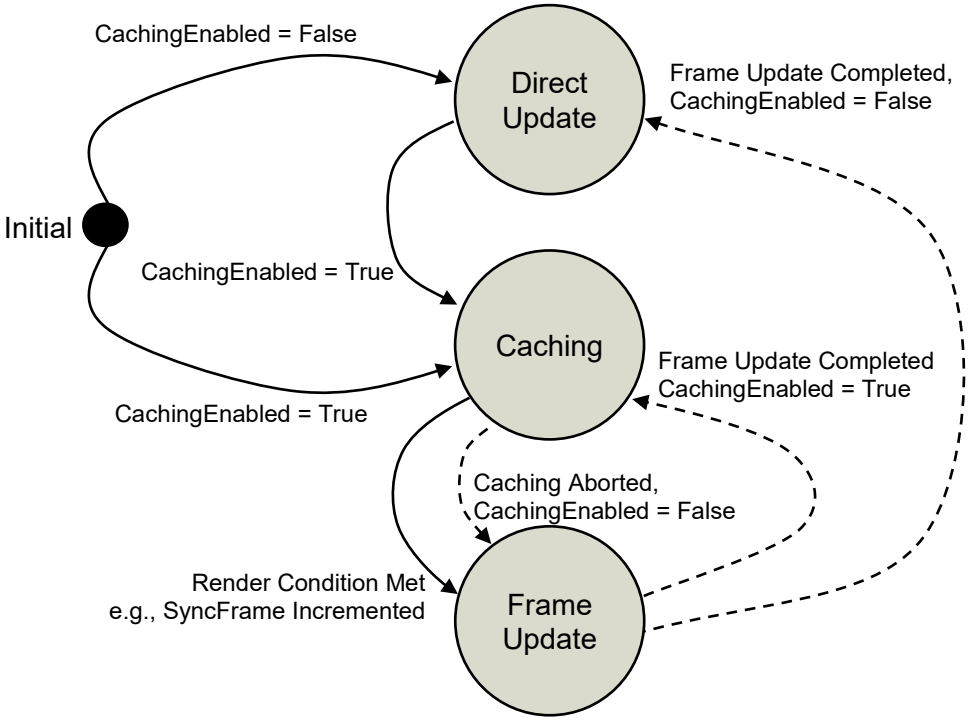


Figure 3.9.18-1 –State Machine for Framing Refresh Container

The DirectUpdate/Caching state is controlled by the CachingEnabled parameter.

On entering the Caching state, the displayed widget tree's current property values are taken as the start point for the cache, all subsequent set parameter commands to this widget's descendants are consumed by the CDS and applied to the cache but not rendered to the display, this continues until the SyncFrame is updated (or CachingEnabled is set to A661_FALSE).

COMMENTARY

FramingRefreshContainer provides a mechanism for the UA to ensure that large volumes of data that are displayed together are always self-consistent.

FramingRefreshContainer, is a widget control approach to the synchronization problem that is portable (as it only requires normal ARINC661 Protocol message operations).

Any limit on the number or complexity of descendent widgets is implementation dependent.

For interactive widgets hosted by a FramingRefreshContainer the SyncFrame may be used (e.g., instead of ContextNumber) to represent the interactive system context.

Restriction:

- A FramingRefreshContainer may not contain another FramingRefreshContainer within its descendants (including via a connector)

FramingRefreshContainer Parameters are defined in Table 3.9.18-1.

Table 3.9.18-1 –FramingRefreshContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_FRAMING_REFRESH_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's descendants to be interactive
Specific parameters		
CachingEnabled	DR	Controls whether or not Frame based updates are active.
SyncFrame	R	The number of the frame currently being displayed, default 0. Represents a UA context.

FramingRefreshContainer Creation Structure is defined in Table 3.9.18-2.

Table 3.9.18-2 – FramingRefreshContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Description
WidgetType	ushort	16	A661_FRAMING_REFRESH_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
CachingEnabled	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	24	0

Available SetParameter identifiers and associated data structure are defined in Table 3.9.18-3.

Table 3.9.18-3 – FramingRefreshContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
CachingEnabled	uchar	8	A661_CACHING_ENABLE
SyncFrame	ushort	16	A661_SYNC_FRAME

3.10 Widgets Added for Supplement 7

3.10.1 ScaleContainer

Categories:

- **Container**

Description:
A ScaleContainer widget applies a scaling transformation to a group of widgets. The ScaleContainer scales its children according to the ScaleX and ScaleY parameters, and the position of its Pivot Center of Scaling.

ScaleX and ScaleY parameters should be positive values.

Note that:

- **Scaling texts is implementation-dependent**
- **CenterX and CenterY only apply to the container itself, not the children positions**

Examples with a ScaleX and ScaleY parameters = 2 :

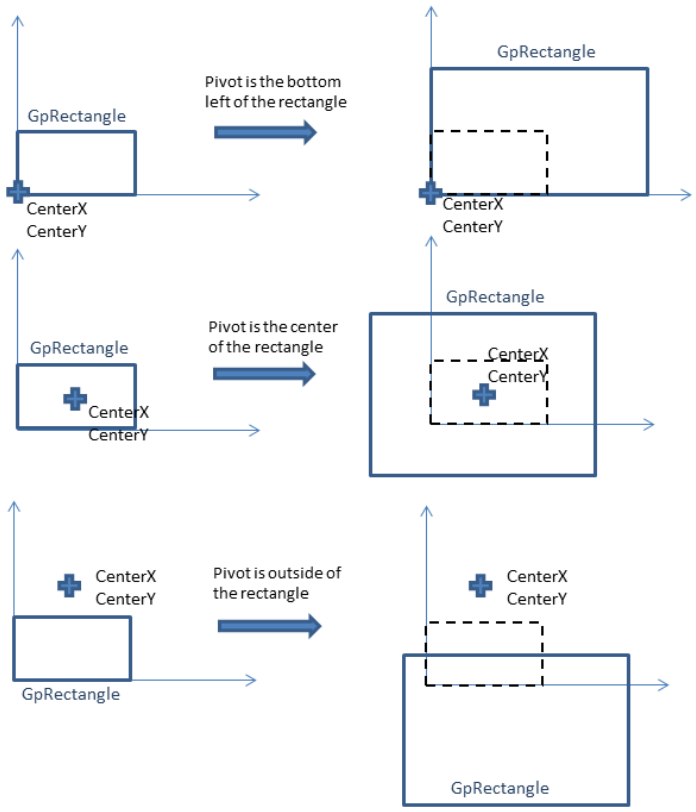


Figure 3.10.1 – ScaleContainer Examples

Restriction:
None

The possible children of this widgets are the same as for the TranslationContainer widget.

Note: The ScaleX and ScaleY parameters must be > 0

ScaleContainer Parameters are defined in Table 3.10.1-1.

Table 3.10.1-1 – ScaleContainer Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SCALE_CONTAINER
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Specific parameters		
CenterX	DR	X coordinate of the scaling Pivot point
CenterY	DR	Y coordinate of the scaling Pivot point
ScaleX	DR	X Scale of the children widgets
ScaleY	DR	Y Scale of the children widgets

ScaleContainer Creation Structure is defined in Table 3.10.1-2.

Table 3.10.1-2 – ScaleContainer Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SCALE_CONTAINER
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
CenterX	long	32	
CenterY	long	32	
ScaleX	float	32	
ScaleY	float	32	

The ScaleContainer widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.10.1-3.

Table 3.10.1-3 – ScaleContainer Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
ScaleX, ScaleY	float x 2	32 x 2	A661_SCALE_XY
ScaleX	float	32	A661_SCALE_X
ScaleY	float	32	A661_SCALE_Y
CenterX, CenterY	long x 2	32 x 2	A661_CENTER_XY
CenterX	long	32	A661_CENTER_X
CenterY	long	32	A661_CENTER_Y

3.10.2 Selector

Categories:

- Interactive
- Graphical representation
- Text string

Description:

The Selector widget allows the user to select a single entry from a set of entries displayed according to the specified graphical layout.

The following figure is an example of a horizontal Selector displaying four entries, having its first entry (labeled “E1”) selected.

Figure 3.10.2-1 – Selector Example

Standardized layouts types are:

- A661_VERTICAL, default top to bottom
- A661_HORIZONTAL, left to right
- A661_CIRCULAR, clockwise, first entry top
- A661_GRID, row major, left to right, top to bottom
- A661_STAR, clockwise, first entry at center, second entry top

Examples of Star layout (1, 2, 3, 4, and 5 entries):

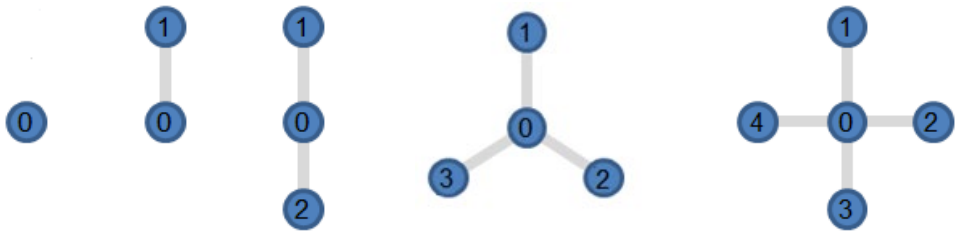


Figure 3.10.2-2 – Examples of Star Layout (1, 2, 3, 4 and 5 entries)

Examples of Circular layout (1, 2, 3, and 4 entries):

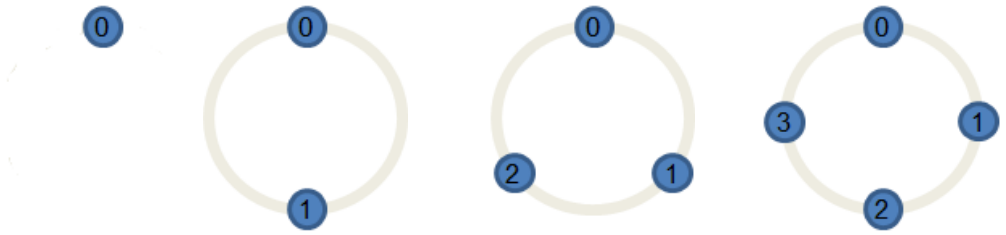


Figure 3.10.2-3 – Examples of Circular Layout (1, 2, 3, and 4 entries)

Selector Parameters are defined in Table 3.10.2-1.

Restriction:

None

Table 3.10.2-1 – Selector Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_SELECTOR
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 (whole widget ability)
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	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter.
Specific parameters		
MaxNumberOfEntries	D	Max number of entries to be managed by the CDS
NumberOfEntries	DR	Number of displayed entries
SelectedEntry	DR	Current selected entry number from 1 to NumberOfEntries if an entry is selected and 0 else.
MaxStringLength	D	Maximum size in bytes (including the NULL terminator) of each string array entry.
Alignment	DR	Alignment of the text for each entry.
Layout	DR	Type of layout (appearance to be defined by the aircraft OEM). Fine tuning done thanks to styleset.
EnableArray	DR	Ability for each entry to be selected in the Selector
StringArray	DR	Strings attached to each entry.
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing.

SelectorCreation Structure is defined in Table 3.10.2-2.

Table 3.10.2-2 – Selector Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_SELECTOR
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
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
Layout	uchar	8	A661_VERTICAL A661_HORIZONTAL A661_CIRCULAR A661_GRID A661_STAR
UnusedPad	N/A	8	0
EnableArray[NumberOf Entries]	{uchar}+	8* NumberOfEntries	
StringArray[NumberOf Entries]	{String}+	8* string length + PAD	

Selector Event Structures: A661_EVT_SEL_ENTRY_CHANGE is defined in Table 3.10.2-3.

Table 3.10.2-3 – Selector Event Structures: A661_EVT_SEL_ENTRY_CHANGE

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

Available SetParameter identifiers and associated data structure are defined in Table 3.10.2-4.

Table 3.10.2-4 – Selector Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
Layout	uchar	8	A661_LAYOUT
EnableArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_ENABLE_ENTRY_ARRAY
StringArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY
StringArray [up to MaxNumberOfEntries] and EnableArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_ENTRY_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
Alignment	**uchar**	**8**	**A661_ALIGNMENT**

3.10.3 Tree

Categories:

- **Interactive**
- **Graphical representation**

Description:
A Tree widget is a set of items organized as a tree. A Tree widget behaves similarly to the ScrollList widget, except that the entries are grouped hierarchically.

The ParentArray identifies the direct parent of each entry. A ParentArray entry must have a value from zero to NumberOfEntries (inclusive) where zero indicates the entry does not have a direct parent and is a root node. At least one ParentArray entry must have a value of zero. A ParentArray entry may not refer to itself as its own parent. Note that all the children of a parent entry must always be after the parent in the array of entries.

Figure 3.10.3-1 shows two trees, one with a single root node and one with multiple root nodes.

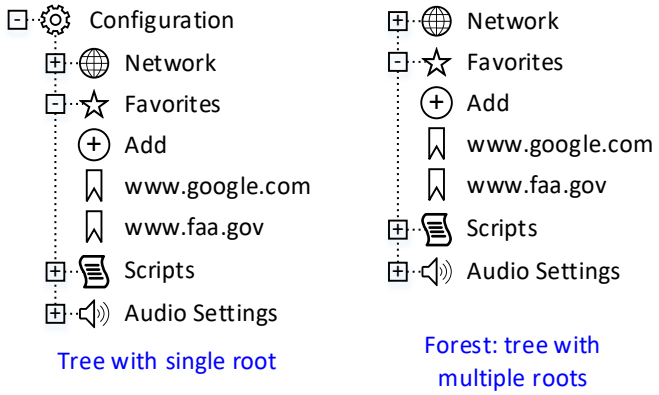


Figure 3.10.3-1 – Single Root and Multiple Root Tree Samples

The displayed order of siblings, that is entries whose ParentArray all have the same value, will be determined by the entry's position in the array. Entries at lower index position will be displayed before entries at large index positions.

The picture or symbol attached to every entry, depending on if the entry is opened or closed, is a parent or a leaf, and has children or not is specified by the NodeStyleArray parameter. The effective picture or symbol to use is implementation dependent.

The visual layout of the tree is implementation dependent. A CDS implementation could display the data using a vertical format (as illustrated in Figure 3.10.3-1) or in a horizontal format (as illustrated in Figure 3.10.3-2). The CDS implementation may support one or the other or both under the control of the widget's StyleSet value.

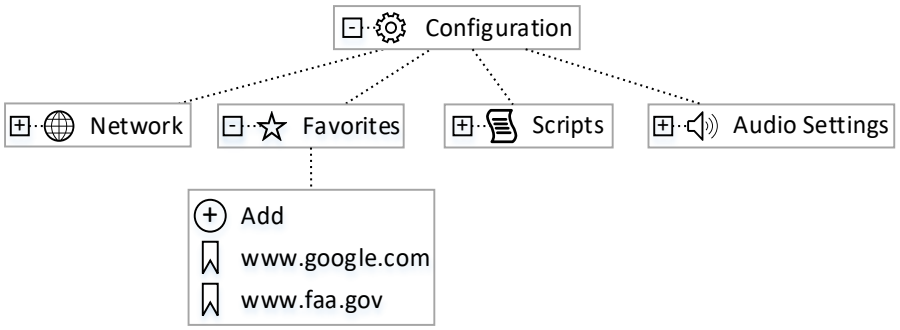


Figure 3.10.3-2 – Possible Horizontal Tree Layout

Restriction: None

Tree Parameters are defined in Table 3.10.3-1.

Table 3.10.3-1 – Tree Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_TREE
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	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
NextFocusedWidget	DR	Widget ident of next widget to be focused. See Section 3.1.3.5.
AutomaticFocusMotion	DR	Automatic motion of the focus on widget specified in NextFocusedWidget parameter
Specific parameters		
FirstVisibleEntry	DR	Index into LabelStringArray/EnableArray of the entry appearing as the first entry in the visible area of the ScrollList
FlagReportVisibleEntry	D	If True, CDS will report change on first visible entry following crew member actions.
MaxNumberOfEntries	D	Max number of entries to be managed by the CDS
NumberOfEntries	DR	Number of entries
SelectedEntry	DR	Current selected entry number from 1 to NumberOfEntries if an entry is selected and 0 else.
MaxStringLength	D	Maximum size in bytes (including the NULL terminator) of each string array entry.
Alignment	DR	Alignment of the text within the label area of an entry LEFT RIGHT CENTER
NodeStyleArray	DR	Array of node styles attached to each entry
EnableArray	DR	Ability for each entry to be selected in the tree: A661_TRUE, A661_FALSE
OpenArray	DR	Indicates for each entry if the entry is opened (A661_TRUE) or closed (A661_FALSE). If the entry is a leaf, the element value is not used
LabelStringArray	DR	The array of Strings attached to each entry.
ParentArray	DR	The parent entry attached to each entry. The value must be from zero to NumberOfEntries (inclusive) where zero indicates the entry does not have direct parent and is a root node. At least one ParentArray entry must have a value of zero.
EntryValidation	R	Indicator notifying the CDS that the UA has completed processing the entry or selection event. This flag also indicates the results of that processing. A661_FALSE A661_TRUE

Tree Creation Structure is defined in Table 3.10.3-2.

Table 3.10.3-2 – Tree Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_TREE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE A661_TRUE_WITH_VALIDATION
Visible	uchar	8	A661_FALSE A661_TRUE
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
FirstVisibleEntry	ushort	16	
FlagReportVisibleEntry	uchar	8	A661_FALSE A661_TRUE
UnusedPad	uchar	8	N/A
StyleSet	ushort	16	
NextFocusedWidget	ushort	16	
MaxNumberOfEntries	ushort	16	
NumberOfEntries	ushort	16	
SelectedEntry	ushort	16	
MaxStringLength	ushort	16	
AutomaticFocusMotion	uchar	8	A661_FALSE A661_TRUE
Alignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
NodeStyleArray[NumberOfEntries]	{ushort}+	16* NumberOfEntries	
ParentArray[NumberOfEntries]	{ushort}+	16* NumberOfEntries	
EnableArray[NumberOfEntries]	{uchar}+	8* NumberOfEntries	
OpenArray[NumberOfEntries]	{uchar}+	8* NumberOfEntries + Pad	
LabelStringArray[NumberOfEntries]	{String}+	8* string length* NumberOfEntries + Pad	

Selector Event Structures: A661_EVT_SEL_ENTRY_CHANGE is defined in Table 3.10.3-3.

Table 3.10.3-3 – Selector Event Structures: A661_EVT_SEL_ENTRY_CHANGE

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

Selector Event Structures: A661_EVT_FIRST_VIS_ENTRY_CHANGE is defined in Table 3.10.3-4.

Table 3.10.3-4 – Selector Event Structures: A661_EVT_FIRST_VIS_ENTRY_CHANGE

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

Selector Event Structures: A661_EVT_TREE_ENTRY_OPEN is defined in Table 3.10.3-5.

Table 3.10.3-5 – Selector Event Structures: A661_EVT_TREE_ENTRY_OPEN

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_TREE_ENTRY_OPEN
SelectedEntry	ushort	16	Index of the tree entry opened by the user

Selector Event Structures: A661_EVT_TREE_ENTRY_CLOSE is defined in Table 3.10.3-6.

Table 3.10.3-6 – Selector Event Structures: A661_EVT_TREE_ENTRY_CLOSE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_TREE_ENTRY_CLOSE
SelectedEntry	ushort	16	Index of the tree entry closed by the user

Available SetParameter identifiers and associated data structure are defined in Table 3.10.3-7.

Table 3.10.3-7 – Tree Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
Enable	uchar	8	A661_ENABLE
StyleSet	ushort	16	A661_STYLE_SET
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
Size X, Size Y	ulong x 2	32 x 2	A661_SIZE_XY
NumberOfEntries	ushort	16	A661_NUMBER_OF_ENTRIES
SelectedEntry	ushort	16	A661_SELECTED_ENTRY
FirstVisibleEntry	ushort	16	A661_FIRST_VISIBLE_ENTRY
EnableArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_ENABLE_ENTRY_ARRAY
OpenArray[up to MaxNumberOfEntries]	N/A	{32}+	A661_OPEN_ENTRY_ARRAY
ParentArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_PARENT_ENTRY_ARRAY
LabelStringArray [up to MaxNumberOfEntries]	N/A	{32}+	A661_STRING_ARRAY
NodeStyleArray[up to MaxNumberOfEntries]	N/A	{32}+	A661_TREE_NODE_STYLE_ARRAY
EntryValidation	uchar	8	A661_ENTRY_VALID
Alignment	uchar	8	A661_ALIGNMENT

3.10.4 MapExternalSource

Categories:

None

Description:

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

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

Note: the MapExternalSource widget makes it necessary for the UA supplier and the integrator to define the specific method to bring video to the display.

The way the video signal input or bitmap image is adapted to the Map characteristics is implementation dependent.

The CDS adapt the source content to present it in the MapHorz or MapVert widget, depending on the MapHorz or MapVert projection reference point, range, and orientation. The projection parameters of the MapHorz or MapVert widget may be sent by the MapExternalSource to the external source through a back-channel. The external source sends back a video signal stream or bitmap image to the MapExternalSource, taking into account these characteristics.

The way the CDS perform this adaptation to ensure consistency between the Map content and the external source presentation is implementation-dependent.

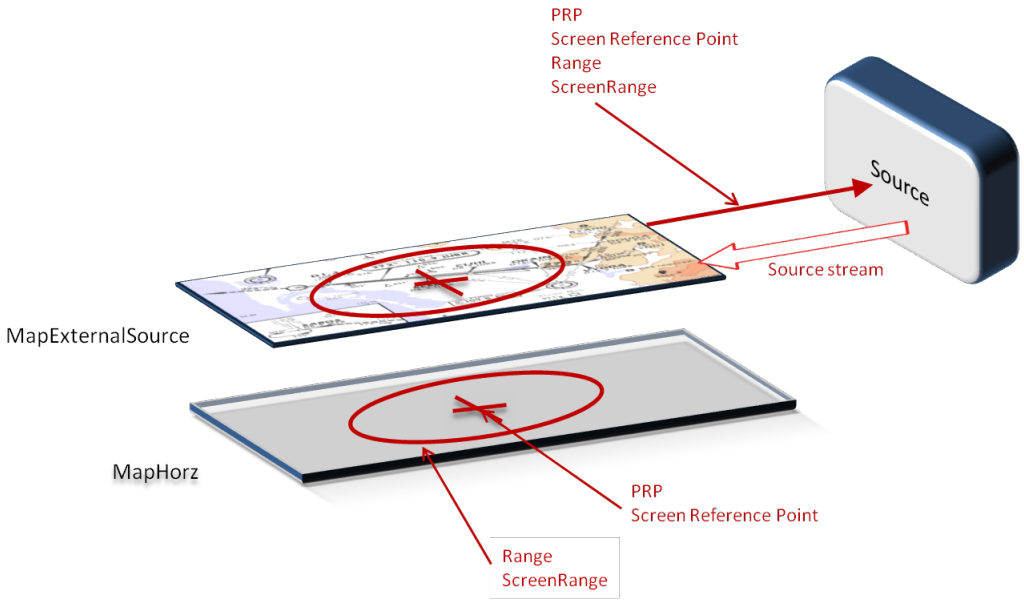


Figure 3.10.4-1 – MapExternalSource Usage in a MapHorz

Some use cases for this widget may be:

- **Presentation of a cartography background under a horizontal Map**
- **Presentation of Radar weather information under a horizontal Map**
- **Presentation of terrain information under a vertical Map**

Note:

1. **The stream content may be clipped to the size of the mapHorz or mapVert parent**
2. **The PosX, PosY, SizeX, SizeY positions and sizes are defined in 100th of mm. They specify the screen size of the Map external source widget inside the MapHorz or MapVert widget.**

Example:

In the following example, we show a digital Map (cartography background) under a horizontal Map.

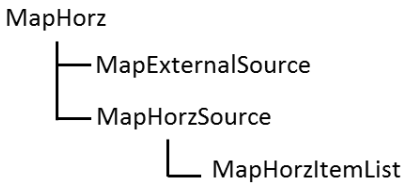


Figure 3.10.4-2 – MapExternalSource Digital Map Example

The size and position of the digital Map covers the entire MapHorz surface, so we have:

- **MapExternalSource.PosX = 0,**
- **MapExternalSource.PosY = 0,**
- **MapExternalSource.SizeX = MapHorz.SizeX,**

- MapExternalSource.SizeY = MapHorz.SizeY

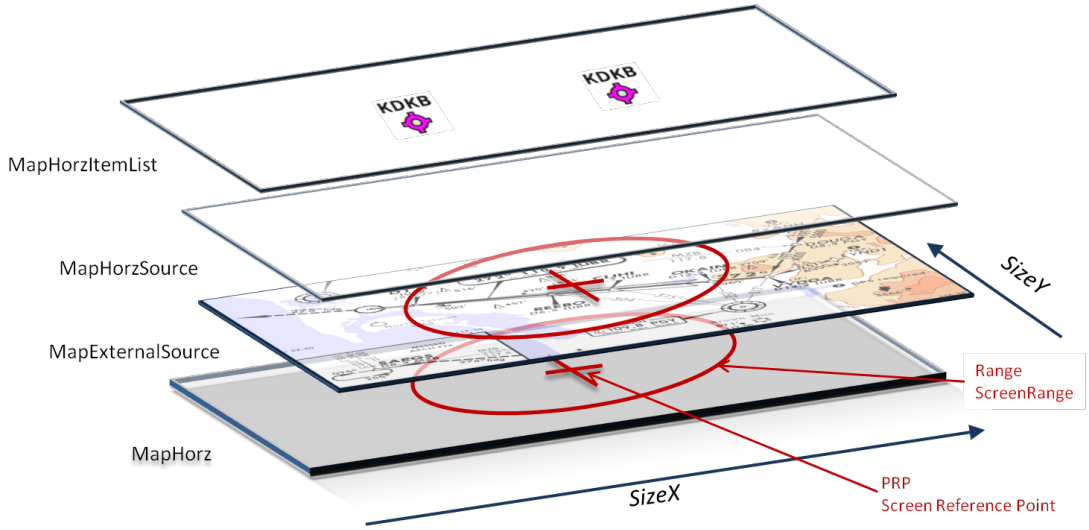


Figure 3.10.4-3 –

Restriction:
None

MapExternalSource Parameters are defined in Table 3.10.4-1.

Table 3.10.4-1 – MapExternalSource Parameters

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_MAP_EXTERNAL_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
PosX	DR	The X position of the widget reference point
PosY	DR	The Y position of the widget reference point
SizeX	DR	The X dimension size (width) of the widget
SizeY	DR	The Y dimension size (height) of the widget
Specific parameters		
SourceReference	DR	Identifier of input stream source reference available on the CDS and used by the UA.
StyleSet	DR	May control transparency, zoom/clip, etc.

MapExternalSource Creation Structure is defined in Table 3.10.4-2.

Table 3.10.4-2 – MapExternalSource Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_MAP_EXTERNAL_SOURCE
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Visible	uchar	8	A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
PosX	long	32	
PosY	long	32	
SizeX	ulong	32	
SizeY	ulong	32	
SourceReference	ushort	16	
StyleSet	ushort	16	

No event is associated with the MapExternalSource widget.

Available SetParameter identifiers and associated data structure are defined in Table 3.10.4-3.

Table 3.10.2-3 – MapExternalSource Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
PosX	long	32	A661_POS_X
PosY	long	32	A661_POS_Y
PosX, PosY	long x 2	32 x 2	A661_POS_XY
SizeX	ulong	32	A661_SIZE_X
SizeY	ulong	32	A661_SIZE_Y
SizeX, SizeY	ulong x 2	32 x 2	A661_SIZE_XY
SourceReference	ushort	16	A661_SOURCE_REF
StyleSet	ushort	16	A661_STYLE_SET

3.10.5 GpTriangleFan

Categories:

- Graphical Representation

Description:
The graphical primitive GpTriangleFan widget enables the definition of a shape composed out of a fan of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the first and last vertices of the previous triangular section. At least three vertices must be specified. Any convex polygon can be represented as a triangle fan, by just specifying its vertices in the natural order. A triangle fan is not necessarily a convex polygon, though.

The following figure shows how the Vertices (V0 ..V6) of a triangle fan define a filled shape formed out of triangles. When drawn, a triangle fan does not draw the interior lines or vertex labels shown in this figure. In this case, it defines a concave polygon:

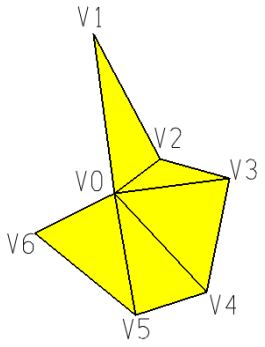


Figure 3.10.5-1 – Triangle Fan – Concave Polygon

The following figure illustrates a convex polygon case:

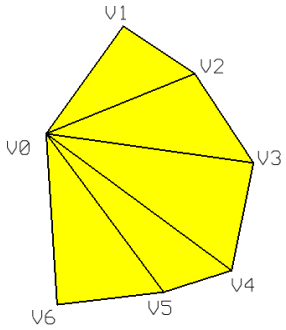


Figure 3.10.5-2 – Triangle Fan – Convex Polygon

Restriction:
None

GpTriangleFan Parameters are defined in Table 3.10.5-1.

Table 3.10.5-1 – GpTriangleFan Parameters

Parameters	Change	Description
Commonly Used Parameters		
WidgetType	D	A661_GP_TRIANGLE_FAN
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 boundary line, see Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of the TriangleFan will be filled.
FillIndex	DR	Fill Pattern index see Section 3.1.3.3.1.
MaxNumberOfVertices	D	The maximum number of vertices to be defined for the widget.
NumberOfVertices	DR	The number of vertices currently defined for the widget.
Vertices	DR	The array of vertices as (X,Y) pairs.

GpTriangleFan Creation Structure is defined in Table 3.10.5-2.

Table 3.10.5-2 – GpTriangleFan Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_TRIANGLE_FAN
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
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 A661_SAME_LEVEL
UnusedPad	N/A	16	0
MaxNumberOfVertices	ushort	16	Must be greater than or equal to 3.
NumberOfVertices	ushort	16	Must be greater than or equal to 3, and must be less than or equal to MaxNumberOfVertices.
Vertices	array of (long, long)	64 * NumberOfVertices	There are NumberOfVertices of (x, y) pairs.

The GpTriangleFan does not send any event.

Available SetParameters identifiers and associated data structure are defined in Table 3.10.5-3.

Table 3.10.5-3 – GpTriangleFan Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Halo	uchar	8	A661_HALO
Filled	uchar	8	A661_FILLED
NumberOfVertices	ushort	16	A661_NUMBER_OF_ENTRIES
Vertices [up to NumberOfVertices]	long x 2	{64}+	A661_VERTICES_ARRAY

3.10.6 GpTriangleStrip

Categories:

- Graphical Representation

Description:
The graphical primitive GpTriangleStrip widget enables the definition of a shape composed out of a linked strip of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the last two vertices of the previous triangular section.

The following figure shows how the Vertices (V0 ..V5) of a triangle strip define a filled shape formed out of triangles. When drawn, a triangle strip does not draw the interior lines or vertex labels shown in this figure.

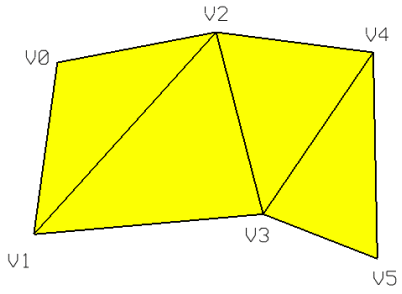


Figure 3.10.6 – Triangle Strip

GpTriangleStrip Parameters are defined in Table 3.10.6-1.

Table 3.10.6-1 – GpTriangleStrip Parameters

Parameters	Change	Description
Commonly Used Parameters		
WidgetType	D	A661_GP_TRIANGLE_STRIP
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 boundary line, see Section 3.1.3.3.1.
Halo	DR	Halo is a full outline in a contrasting color (typically black) to enhance readability. See Section 3.1.3.3 for a description of possible values.
Filled	DR	If set to True, interior of TriangleStrip will be filled.
FillIndex	DR	Fill Pattern index see Section 3.1.3.3.1.
MaxNumberOfVertices	D	The maximum number of vertices to be defined for the widget.
NumberOfVertices	DR	The number of vertices currently defined for the widget.
Vertices	DR	The array of vertices as (X,Y) pairs.

GpTriangleStrip Creation Structure is defined in Table 3.10.6-2.

Table 3.10.6-2 – GpTriangleStrip Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_GP_TRIANGLE_STRIP
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Anonymous	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE
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 A661_SAME_LEVEL
UnusedPad	N/A	16	0
MaxNumberOfVertices	ushort	16	Must be greater than or equal to 3.
NumberOfVertices	ushort	16	Must be greater than or equal to 3, and must be less than or equal to MaxNumberOfVertices.
Vertices	array of (long,long)	64 * NumberOfVertices	There are NumberOfVertices of (x,y) pairs.

GpTriangleStrip does not send any event.

Available SetParameters identifiers and associated data structure are defined in Table 3.10.6-3.

Table 3.10.6-3 – GpTriangleStrip Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
Visible	uchar	8	A661_VISIBLE
StyleSet	ushort	16	A661_STYLE_SET
ColorIndex	uchar	8	A661_COLOR_INDEX
FillIndex	uchar	8	A661_FILL_INDEX
Halo	uchar	8	A661_HALO
Filled	uchar	8	A661_FILLED
NumberOfVertices	ushort	16	A661_NUMBER_OF_ENTRIES
Vertices [up to NumberOfVertices]	long x 2	{64}+	A661_VERTICES_ARRAY

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, the CDS processes one or more Definition Files (DF) for each UA. At run-time, messages are exchanged between UAs and CDS.

4.2 Definition Phase Exchange

4.2.1 Definition File and UALD

A DF contains User Application Layer Definitions (UALDs) from one UA. The UALD describes data shared between the UA and the CDS, as depicted in Figures 4.2.2a and 4.2.2b

The DF contains a header and footer. 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 (see Section 4.5.2). Each block (i.e., each UALD) contains the definition of exactly one layer. Support for multiple blocks in a file is implementation dependent.

Each UA displaying data inside the CDS is associated with at least one layer. The UALD describes the hierarchical structure of the UA widgets residing in the layer as well as the specific ARINC 661 interface parameters of these widgets. The format of a block is described in Sections 3.0 and 4.5. The hierarchical structure of the widgets is ensured by the use of the “ParentIdent” parameter in each widget definition.

For each type of widget, all parameters must have a CDS default value. If a parameter is not set in the UALD, the CDS default value will be used. This will happen in the case of run-time-only parameters, or in the case where a CDS library definition has been updated to incorporate new parameters, but an older UALD is not updated.

Figure 4.2.2a shows widgets “owned” by the UA (primarily, defines their IDs) for display of information on CDS. This is the definition of the static graphical part of the UA interface for one layer.

Typically, a CDS will load the DF containing the UALDs directly from a data store as shown in Figure 4.4.2b. The CDS interprets the UALD data to allocate and construct the hierarchical tree of widgets instances in conjunction with the CDS widget library.

At run-time the CDS and UA exchange ARINC 661 messages to manage the widget parameters, and thus their graphical representation.

4.2.2 Binary Format

The DF describes shared data. Therefore, one main objective of this specification is to standardize the DF data and format shared between the UA and the CDS. The graphical part of the application interface (DF) must be loaded into the CDS. This DF describes only data, and is therefore interpreted by the CDS to be associated with the CDS graphical capacities.

To limit the impact of a UA modification on the CDS, the data format loaded into CDS should be independent from the CDS. For growth potential, a data format independent from the CDS should provide the capability to UAs to download their

Definition Files (DF) into the CDS. The format for the DF is a standard non-executable binary format, allowing uploading and downloading of the DF.

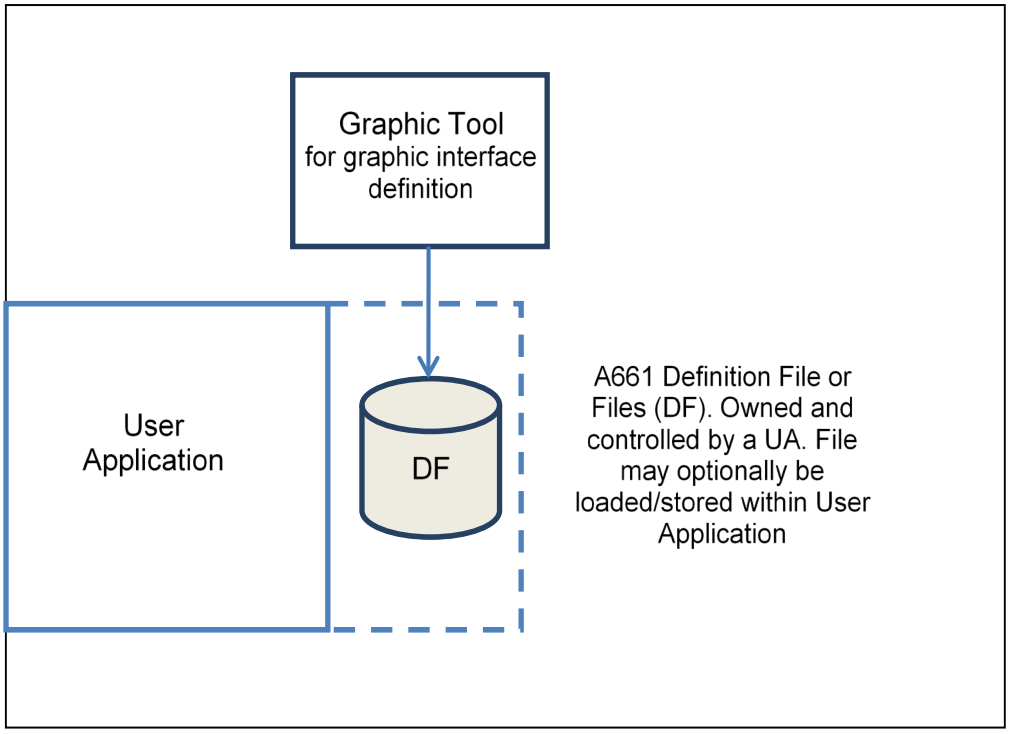


Figure 4.2.2a – UA DF Generation and Ownership

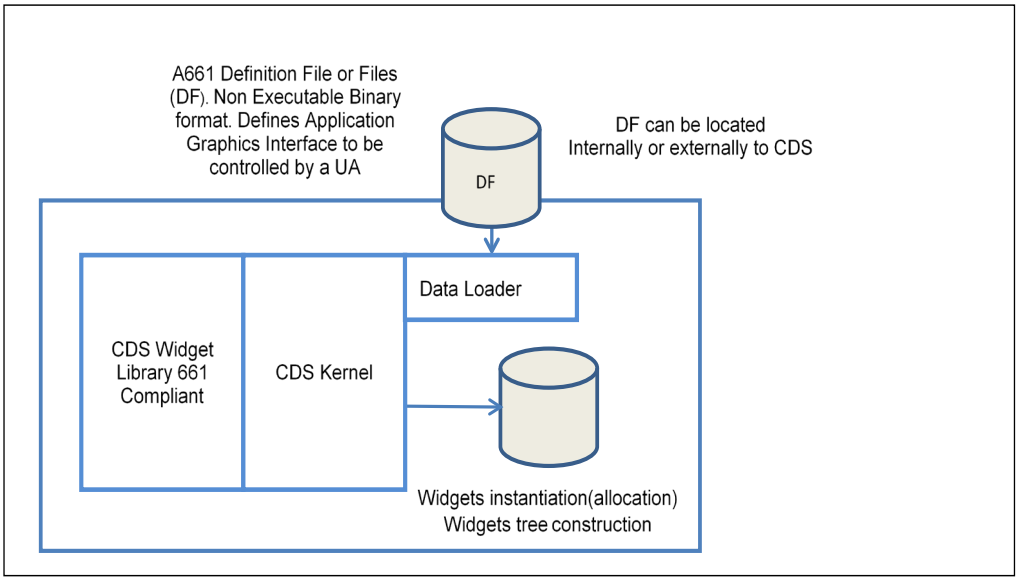


Figure 4.2.2b – Example CDS Definition File Relationship

4.3 Run-time Communication

4.3.1 General Principle

The communication from UA to CDS concerns run-time widget parameters for widgets management. The transmittal of these parameters corresponds to:

A context change of the application, which could correspond to current periodic parameter transmittal

- A response to a widget event, which is purely asynchronous

The communication from CDS to UA is event driven. The transmittal of these parameters corresponds to:

- Event notification from a widget, which is purely asynchronous

Basically, the communication from UA to CDS is event driven. Therefore, all messages will be sent in an asynchronous way. Asynchronous exchange for UA functional context change should save bandwidth as well as decrease latency-time values. Nevertheless, if synchronous refresh of information is needed for a particular UA, the UA could send the information on a periodic basis.

Due to the Client-Server architecture of ARINC 661, a communication channel between a CDS and a User Application is logically a point-to-point connection, even if the actual network architecture uses multicast or broadcast mechanisms.

COMMENTARY

Concerning widget states, the associated parameters should be managed in an asynchronous way. Periodic transmittal of messages for controlling widget states could raise a “race condition,” described in Appendix A, Glossary.

4.3.2 Issues

The asynchronous approach should mitigate the following design concerns:

1. Loss of a message: From UA to CDS, the display will not be in the state that the UA expects. From CDS to UA, the UA will not react to a crew member interaction. Refer to Section 4.3.3, Assumption on Communication Reliability.
2. Synchronization: A UA needs to be sure that a message will arrive before the next transmitted message. Refer to Section 4.3.3, Assumption on Communication Reliability.
3. Reconfiguration of the rendering unit: The new rendering subsystem may not have needed information available. Refer to Section 2.3.2.4, Layer Activity/Visibility Management.

4.3.3 Assumption on Communication Reliability

This specification is independent of any bus choice. Nevertheless, some assumption should be made on the level of reliability. Therefore, the basic assumption is that the communication is in reliable packet order.

This applies to:

- The correct reception order of messages (see Issue 2 above)
- The loss of messages (see Issue 1 above)

4.3.4 Layer Data Management

Refer to Section 2.3.2.4, Layer Activity/Visibility Management.

4.4 ARINC 661 Commands

4.4.1 Type of Commands

Table 4.4.1-1 describes the definition-time command.

Table 4.4.1-1 – Definition-Time Command

	Command Type	Description
UA ⇓ CDS	A661_CMD_CREATE	Creation of a widget with definition of its parameters. For a version of the Widget library, all the parameters of a used widget should be set. Refer to Section 4.5, ARINC 661 Command Structure. Parameter order inside the Creation buffer is provided in each widget description in Section 3.3, Widget List.
UA ⇓ CDS	A661_CMD_EXTEND	Extension of a previously defined widget with definition of the parameters necessary for the new function associated with the extension. Refer to Section 4.5.3.3.2, Extension Command Structure. Parameter order inside the Extension buffer is provided for each Widget Extension in Section 8, Widget Extensions.

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 4.4.3.1, Request from UA to CDS).
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, Notification from CDS to UA).
	A661_NOTIFY_EXCEPTION	Notification of an error from CDS to UA. Refer to Section 4.4.2, Error Notification, for all applicable errors.

4.4.2 Error Notification

This specification defines the principle of error notification to provide the tools to manage errors in a command. Nevertheless, it is outside the scope of this document to define the recovery action on an error notification. The aircraft OEM should specify only error recording in Built-In Test Equipment (BITE), or some recovery mechanism to implement specific errors.

The error notifications which the CDS may send in response to erroneous commands and overload situations are described in Table 4.4.2 in this specification.

This ARINC specification lists valid values that User Applications and the CDS are supposed to use when exchanging run-time messages. The CDS's response to undefined values is implementation dependent. At the CDS's discretion, error messages may be created and sent to the User Application if, for example, a value other than "0" and "1" is sent as a Boolean parameter value, or the CDS may choose to fall back to a default value (for example, interpret every non-zero valid the same as A661_True, or only look at the least significant bit of the 8-bit value). This applies analogous to some other parameter types, such as those using enumerated types. Note that this is not to be understood as an encouragement for a sender of 661 run-time messages to rely on the behavior of a specific CDS implementation when sending data to that CDS, because doing so could lead to an undesirable dependency between a User Application and that CDS.

Any or all of the following notifications are sent at the discretion of the CDS.

Table 4.4.2 – Exception Type

Command/ Request type	A661_ExceptionType	Description
Error notification on command error		
All	A661_ERR_BAD_COMMAND	This exception is sent on any erroneous command. It applies to: Invalid block structure (keyword or size field) Invalid command/request code
Create	A661_ERR_CREATE_ABORTED	This exception is sent on erroneous Create command. It applies to: Invalid Layer ID or Context ID Invalid Widget ID Invalid parameter value Insufficient required parameter data (growth potential for direct downloading the DF from UA to CDS)
Extend	A661_ERR_EXTEND_ABORTED	This exception is sent on erroneous Extend command. It applies to: Invalid Extension ID Invalid parameter value Insufficient required parameter data Extension applied to invalid widget type
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

Command/ Request type	A661_ExceptionType	Description
UARequest	A661_ERR_UA_REQUEST_ABORTED	This exception is sent on erroneous UA Request. It applies to: Invalid Layer ID or Context ID Invalid Request key value Invalid Widget ID Insufficient required parameter data
Error notification on CDS Resource overload		
	A661_ERR_MEMORY_OVERLOAD	Notification of memory overloading by allotting UA widgets. (Definition time). (Growth potential for direct downloading the DF from UA to CDS)
	A661_ERR_PROCESS_OVERLOAD	Inability to complete processing of desired image.
	A661_ERR_RENDERING_OVERLOAD	Inability to complete rendering of desired image.

Because the level of CDS is the higher application, there is no need for exceptions on commands from CDS to UA.

4.4.3 ARINC 661 Request/Notification

Communication described in this specification is based on the widget management. Requests apply to messages exchanged between UA and the CDS without a particular widget being targeted. These messages provide a means for the UA to change HMI mechanisms under CDS responsibility, such as Focus position or layer activity. In the other direction, these messages provide the CDS a means to indicate the current state to the UA.

4.4.3.1 Request from UA to CDS

Requests from the UA to the CDS, described in Table 4.4.3.1, may or may not be accepted by the CDS. These requests should be sent to the CDS through A661_CMD_UA_REQUEST command.

Table 4.4.3.1 – Request from UA to CDS

Request Type	Description
A661_REQ_LAYER_ACTIVE	Provide a means for a UA application to request to the CDS the activation of its layer. The CDS may or may not accept the request according to the current possible configuration. When a layer is active, the CDS should update widget parameters of this layer (refer to Section 2.3.2.4 – Layer Activity/Visibility Management).
A661_REQ_LAYER_INACTIVE	Provide a means for a UA to request the CDS to deactivate its layer.
A661_REQ_FOCUS_ON_WIDGET	Move focus on a defined widget of one UA layer.
A661_REQ_LAYER_VISIBLE	Turn on the visibility of one layer. This request follows the CDS notification of the layer activity.
A661_REQ_CURSOR_ON_WIDGET	Move the cursor to a defined widget of one UA layer.

4.4.3.2 Notification from CDS to UA

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 take into account the notification.

Table 4.4.3.2 – Notification from CDS to UA

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.
A661_NOTE_CYCLIC_ACTIVATION	Cyclic notification from CDS to the UA that its layer is active. Used for UA to detect CDS failure or loss of connection or recover from a UA failure. Can also be used by CDS to relay new information to UA without going through a deactivation/activation series of notifications (see Reserve data fields in Table 4.5.4.4-5). Transmission rate is implementation dependent.
A661_NOTE_CONNECTOR_DATA	Notification containing information from the DataConnector widget. See widget description in Section 3.9.16 for details.

4.5 ARINC 661 Command Structure

4.5.1 Notation

Relocated to Section 3.3 by Supplement 6.

4.5.2 Block Structure

The UA and CDS exchange information through blocks of commands. A block represents a set of data to be processed as coherent information. Blocks can be exchanged at Definition Time and during Runtime.

The structures detailed in Sections 4.5.3 and 4.5.4 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 Exchange Structure

4.5.3.1 UADF Loading Structure

The Definition File (DF) is recommended to be loadable into CDS according to the recommendations of ARINC report 615A. The format of the loadable software is described in ARINC Report 665. In this case the DF is defined as a data file.

According to ARINC 665, the data file will be a binary file with the file extension “LUP”. The ARINC 665 header file has the file extension “LUH”. The header file defines which data file(s) are to be loaded.

It is recommended that only one data file be attached to the ARINC 665 header of a DF. In this case the “Number of Data Files” field in the ARINC 665 header should be set to one (1).

It is recommended that no support file be attached to the ARINC 665 header of a data file. The “Number of Support Files” field in the ARINC 665 header should be set to zero (0).

4.5.3.2 Definition File (DF) Structure

A Definition File (DF) may hold several User Application Layer Definitions (UALD) from one UA, zero or more Symbol definitions and zero or more Picture definitions. The loadable entity is the DF. Table 4.5.3.2-1 describes the structure of a DF.

Table 4.5.3.2-1 – Definition File Structure

A661_Definition_File	Description
DF_File_Header	Header of the file (see Table 4.5.3.2-2)
[A661_Charset_Encoding_Block_Structure_DT]	Defines the character set encoding for all strings (see Table 4.5.3.2-6)
[A661_Symbol_Block_Structure_DT]	Defines ARINC 661 symbols (see Table 4.5.3.2-5). Prior to Supplement 6, ARINC 661 allowed more than one occurrence; the handling of that case is implementation dependent.
{ A661_Picture_Block_Structure_DT }	Defines ARINC 661 Pictures (bitmaps) (see Tables in Section 7)
[A661_Animation_Law_Block_Structure_DT]	Defines ARINC 661 Animation Laws (see tables in Section 9).
{ A661_Block_Structure_DT }+	Defines ARINC 661 Layers (see Table 4.5.3.2-4)
DF_File_Footer	Footer of the file.(see Table 4.5.3.2-3)

The DF has a header with an implementation dependent part:

Table 4.5.3.2-2 – Definition File Header

DF_File_Header	Type	Size (bits)	Description
A661_DF_MAGIC_NUMBER	ushort	16	Identifier for A661 Definition File
LibraryVersion	uchar	8	Identifier of the Library version compatible with the User Application Definition File. This value is implementation dependent.
SuppVersion	uchar	8	Identifier of the ARINC 661 Supplement version on which the User Application Definition File is based. A661 & A661-1 value = “00” A661-2 value = “02” A661-3 value = “03”, etc.
AppliIdent	ushort	16	Identifies the User Application to which this DF applies (not necessarily [UserApplicationIdent]).
Size of the OEM free data	ushort	16	Size of the OEM_Free_Data in bytes
OEM_Free_Data	N/A	{32}+	

The DF has a footer:

Table 4.5.3.2-3 – Definition File Footer

DF_File_Footer	Type	Size (bits)	Description
A661_DF_FOOTER	uchar	8	Keyword to indicate the end of the Definition File
UnusedPad	N/A	24	0

The DF can contain data blocks defining Layers, Symbols, and Pictures to be exchanged between UA and CDS at definition time. Block structures for Layer and Symbol Blocks appear below. See the tables in Section 7 for Picture Block structure details.

Table 4.5.3.2-4 – Layer Block Structure Exchanged Between UA and CDS at Definition Time

A661_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_LAYER_BLOCK	uchar	8	Start keyword opening a block of information
LayerIdent	uchar	8	Relative Identifier of the layer for the User Application
Context Number	ushort	16	ContextNumber value attached to the layer. This value is attached by the CDS to any block of message sent before communication is established with UA.
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Definition_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_LAYER_BLOCK	uchar	8	Keyword ending a block of information.
UnusedPad	N/A	24	0

Table 4.5.3.2-5 – Symbol Block Structure Exchanged Between UA and CDS at Definition Time

A661_Symbol_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_SYMBOL_BLOCK	uchar	8	Start keyword opening a symbol definition.
UnusedPad	N/A	24	0
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Symbol_Definition_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_SYMBOL_BLOCK	uchar	8	Keyword ending a block of symbol information.
UnusedPad	N/A	24	0

Table 4.5.3.2-6 – Charset Encoding Block Structure Exchanged Between UA and CDS at Definition Time

A661_Charset_Encoding_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_CHARSET_ENCODING_BLOCK	uchar	8	Start keyword opening a charset encoding definition.
UnusedPad	N/A	8	0
A661_Charset_Encoding	ushort	16	Encoding value (see below)
A661_END_CHARSET_ENCODING_BLOCK	uchar	8	Keyword ending a block of charset encoding information.
UnusedPad	N/A	24	0

This encoding applies to all string type parameters in the Definition File, as well as the Run-time Parameters for widgets in the Definition File.

There is only one character set encoding per Definition File.

If this optional Charset Encoding block is not present in the DF, the default A661_ASCII_EXTENDED character set is used.

Possible values for A661_Charset_Encoding are:

- A661_ASCII_EXTENDED: for the ASCII Extended encoding.
- A661_UTF8: for the UTF-8 encoding.
- A661_GBK: for the GBK encoding.

The A661_ASCII_EXTENDED character set case must be supported. Other implementation dependent settings of A661 Charset Encoding are also allowed. Refer to Section 3.2.5.1 for information related to character encodings.

4.5.3.3 Layer Block Definition Commands

One or more A661_Definition_Command can be included in a Layer Block. Table 4.5.3.3 lists the commands defined in the protocol.

Table 4.5.3.3 – Layer Block Definition Commands

A661_Definition_Command	Type	Size (bits)	Description
A661_Create_Structure	N/A	{32}+	Create a new widget.
A661_Extend_Structure	N/A	{32}+	Extend a previously created widget.

4.5.3.3.1 Create Command Structure

Table 4.5.3.3.1 defines the Create command structure.

Table 4.5.3.3.1 – Create 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 descriptions in Section 3 for the description of all the creation parameter buffers.

4.5.3.3.2 Extension Command Structure

Table 4.5.3.3.2 defines the Extension command structure.

Table 4.5.3.3.2 – Extension Command Structure

A661_Extend_Structure	Type	Size (bits)	Description
A661_CMD_EXTEND	ushort	16	Start keyword for opening the extend structure
CommandSize	ushort	16	Size of the command, in bytes.
ExtendParameterBuffer	N/A	{32}+	Refer to Section 8 for the description of all the extend parameter buffers.

4.5.3.3.3 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. This implies that a UALD cannot be described over several LayerBlocks.
- Inside a LayerBlock, a widget should be created (with the A661_CMD_CREATE command, refer to Section 4.4.1) after its parent (as defined by ParentIdent parameter, refer to Section 3.1.3.1).
- For reference from a widget to another widget without ParentIdent, there is no restriction on the definition order. For instance, the ActiveTabbedPanelID parameter of the TabbedPanelGroup will be set before the TabbedPanel is actually referenced in the block. However, the consistency of the DF should be checked.

4.5.3.4 Symbol Block Definition Commands

One or more A661_Symbol_Definition_Commands can be included in a Symbol Block. Table 4.5.3.4 lists the commands defined in the protocol.

Table 4.5.3.4 –Symbol Block Definition Commands

A661_Symbol_Definition_Command	Type	Size (bits)	Description
A661_Create_Symbol_Structure	N/A	{32}+	Create a new symbol.

4.5.3.4.1 Create Symbol Command Structure

Table 4.5.3.4.1 defines the Create Symbol command structure.

Table 4.5.3.4.1 – Create Symbol Command Structure

A661_Create_Symbol_Structure	Type	Size (bits)	Description
A661_CMD_CREATE_SYMBOL	ushort	16	Start keyword for opening the create structure
CommandSize	ushort	16	Size of the command, in bytes.
SymbolId	ushort	16	Symbol ID Value
UnusedPad	N/A	16	0
CreateParameterBuffer	N/A	{32}+	Refer to Section 5 for the description of all the creation parameter buffers

4.5.3.4.2 Constraints Inside a Symbol Definition Block

A symbol can be split into up to four different representations:

- The standard representation
- The focus representation
- The highlight representation
- The sensitive area representation

Refer to Section 5 for a description of the symbol definition commands, as well as any other restrictions on symbol definitions.

4.5.4 Run-Time Exchange Structure

4.5.4.1 Run-Time Block Commands

One or more A661_Run-Time_Command can be included in a block. Run-time structures do not have any implementation dependent headers/footers. They may have bus-specific or network-specific packaging at the transport level.

Table 4.5.4.1-1 – Block Structure Exchanged Between UA and CDS at Run Time

A661_Block_Structure_RT	Type	Size (bits)	Description
A661_BEGIN_BLOCK	uchar	8	Start keyword opening a block of information.
LayerIdent	uchar	8	Relative Identifier of the layer for the User Application
Context Number	ushort	16	ContextNumber value attached to one layer. UA->CDS: Value to be returned by CDS with subsequent blocks. CDS->UA: Value attached by UA on last received block.
Block Size	ulong	32	Size of this block, including header, in bytes.
{ A661_Run-Time_Command }+	N/A	{32}+	Set of command structures, as applicable.
A661_END_BLOCK	uchar	8	Keyword ending a block of information.
UnusedPad	N/A	24	0

Table 4.5.4.1-2 lists commands defined in the protocol:

Table 4.5.4.1-2 – Run-Time Block Commands

A661_Run-Time_Command	Description
A661_Set_Parameter_Structure | A661_Widget_Event_Structure | A661_UA_Request_Structure | A661_CDS_Notification_Structure | A661_Error_Notification_Structure	Commands applicable at run-time.

The following sections define run-time commands and event notifications.

4.5.4.2 Command Structure – Run-Time Commands

Table 4.5.4.2-1 – Set_Parameter_Structure

A661_Set_Parameter_Structure	Type	Size (bits)	Description
A661_CMD_SET_PARAMETER	ushort	16	Start keyword for opening the set parameter structure.
CommandSize	ushort	16	Size of the command, in bytes.
WidgetIdent	ushort	16	Identifier of the widget
UnusedPad	N/A	16	0
{A661_ParameterStructure}+	N/A	{32}+	Set of parameters with the associated values. Refer to Section 4.5.4.5.

Table 4.5.4.2-2 – UA_Request_Structure

A661_UA_Request_Structure	Type	Size (bits)	Description
A661_CMD_UA_REQUEST	ushort	16	Start keyword for opening the UA request structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_Request_Structure	N/A	{32}+	Type of request from UA to CDS. Refer to Section 4.5.4.3.

Table 4.5.4.2-3 – Widget_Event_Structure

A661_Widget_Event_Structure	Type	Size (bits)	Description
A661_NOTIFY_WIDGET_EVENT	ushort	16	Start keyword for opening the widget event structure.
CommandSize	ushort	16	Size of the command, in bytes.
WidgetIdent	ushort	16	Identifier of the widget
EventOrigin	ushort	16	Identifier of the input device which has been used to initiate the event: (Enumeration to be defined by OEM)
EventStructure	N/A	{32}+	Refer to the widget library for the structure of each widget associated events.

Table 4.5.4.2-4 – CDS_Notification_Structure

A661_CDS_Notification_Structure	Type	Size (bits)	Description
A661_NOTIFY_LAYER_EVENT	ushort	16	Start keyword for opening the CDS notification structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_Layer_Notification_Structure	N/A	{32}+	Type of notification from UA to CDS. Refer to Section 4.5.4.4.

Table 4.5.4.2-5 – Error_Notification_Structure

A661_Error_Notification_Structure	Type	Size (bits)	Description
A661_NOTIFY_EXCEPTION	ushort	16	Start keyword for opening the error notification structure.
CommandSize	ushort	16	Size of the command, in bytes.
A661_ExceptionType	ushort	16	Type of error to be notified. Refer to Table 4.4.2.
UnusedPad	N/A	16	0
OEM_free_data	N/A	{32}+	OEM may or may not add free data according to specified mechanism for recovering the error.

4.5.4.3 Request Structure

Table 4.5.4.3-1 – Request_Structure

A661_Request_Structure	Description
A661_Layer_Active_Struct | A661_Layer_Inactive_Struct | A661_Focus_On_Widget_Struct | A661_Layer_Visible_Struct | A661_Cursor_on_Widget_Struct	Type of request

Table 4.5.4.3-2 – Layer_Active_Structure

A661_Layer_Active_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_ACTIVE	ushort	16	Start keyword for opening the layer active structure
UnusedPad	N/A	16	0

Table 4.5.4.3-3 – Layer_Inactive_Structure

A661_Layer_Inactive_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_INACTIVE	ushort	16	Start keyword for opening the layer inactive structure
UnusedPad	N/A	16	0

Table 4.5.4.3-4 – Focus_On_Widget_Structure

A661_Focus_On_Widget_Structure	Type	Size (bits)	Description
A661_REQ_FOCUS_ON_WIDGET	ushort	16	Start keyword for opening the focus on widget structure.
WidgetIdent	ushort	16	Identifier of the widget on which the CDS should move the focus.

Table 4.5.4.3-5 – Layer_Visible_Structure

A661_Layer_Visible_Structure	Type	Size (bits)	Description
A661_REQ_LAYER_VISIBLE	ushort	16	Start keyword for turning on the visibility of one layer.
UnusedPad	N/A	16	0

Table 4.5.4.3-6 – Cursor_On_Widget_Structure

A661_Cursor_On_Widget_Structure	Type	Size (bits)	Description
A661_REQ_CURSOR_ON_WIDGET	ushort	16	Start keyword for opening the cursor on widget structure.
WidgetIdent	ushort	16	Identifier of the widget on which the CDS should move the cursor.
OEM Data Field	N/A	32	This could be used for implementation dependent behavior, e.g., to define which cursor to move

4.5.4.4 Notification Structure

Table 4.5.4.4-1 – Layer_Notification_Structure

A661_Layer_Notification_Structure	Description
A661_Layer_Is_Active_Structure | A661_Layer_Is_Inactive_Structure | A661_Reinitialize_Layer_Data_Structure | A661_Cyclic_Activation_Structure | **A661_Connector_Data_Structure**	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.
Reserved	N/A	16	Reserved for OEM customization

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

Table 4.5.4.4-5 – Cyclic_Activation_Data_Structure

A661_Cyclic_Activation_Structure	Type	Size (bits)	Description
A661_NOTE_CYCLIC_ACTIVATION	ushort	16	Keyword for periodic notification that layer is active.
Reserved	N/A	16	Reserved for OEM customization
Reserved	N/A	32	Reserved for OEM customization

Table 4.5.4.4-6 – Connector_Data_Structure

A661_Connector_Data_Structure	Type	Size (bits)	Description
A661_NOTE_CONNECTOR_DATA	ushort	16	Start keyword for opening the connector data structure.
DataLength	ushort	16	Number of 32-bit words in the Data parameter of the associated DataConnector widget.
Data	N/A	{32}+	Data parameter of the associated DataConnector widget.

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. All the structures are listed here with references as necessary.

4.5.4.5.1 A661_ParameterStructure_1Byte

Table 4.5.4.5.1 – A661_ParameterStructure_1Byte

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

4.5.4.5.2 A661_ParameterStructure_2Bytes

Table 4.5.4.5.2 – A661_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 – A661_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 – A661_ParameterStructure_String

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
String size	ushort	16	Size of the string, in bytes, including terminating NULL.
ParameterValue	string	{32}+	List of char terminated by NULL. Padded by zero, one, two, or three NULL character(s) to be 32 bits aligned

4.5.4.5.5 A661_ParameterStructure_StringArray

Table 4.5.4.5.5 – A661_ParameterStructure_StringArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
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 – A661_StringArray_CellStructure

A661_ParameterStructure	Type	Size (bits)	Description
StringIndex	ushort	16	Index of the string
string size	ushort	16	Integer Size of the string, in bytes, including terminating NULL.
String	string	{32}+	List of char. Ended by one, two, three or four NULL character(s) to be 32 bits aligned

4.5.4.5.7 A661_ParameterStructure_EnableArray

This structure was deprecated by Supplement 4. Replaced by A661_ParameterStructure_1ByteArray.

Table 4.5.4.5.7-1 – A661_ParameterStructure_EnableArray

A661_ParameterStructure	Type	Size (bits)	Description
Parameter_ident	ushort	16	A661_ENABLE_ARRAY
EntryIndex	uchar	8	
Enable	uchar	8	A661_FALSE A661_TRUE

4.5.4.5.8 A661_ParameterStructure_8Bytes

Table 4.5.4.5.8 – A661_ParameterStructure_8Bytes

A661_ParameterStructure_8Bytes	Type	Size (bits)	Description
Parameter_ident	ushort	16	Identifier of the parameter type
UnusedPad	N/A	16	0
ParameterValue1	N/A	32	
ParameterValue2	N/A	32	

4.5.4.5.9 A661_ParameterStructure_BufferOfItems

For usage with MapHorz_ItemList widget, see description in Section 3.3.22.2.2. For usage with MapVert_ItemList widget, see description in Section 3.4.5.2.2.

4.5.4.5.10 A661_ParameterStructure_Buffer

A661_ParameterStructure_Buffer is defined in Table 4.5.4.5.10.

Refer also to BufferFormat widget description in Section 3.3.4 and the BufferFormat data alignment and padding info in Section 3.3.4.1.

Table 4.5.4.5.10 – A661_ParameterStructure_Buffer

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_BUFFER_OF_PARAM
Size	ushort	16	Size of BufferFormatData in bytes.
{BufferFormatData}+	N/A	{32}+	

4.5.4.5.11 A661_ParameterStructure_1ByteArray

Table 4.5.4.5.11 – A661_ParameterStructure_1ByteArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
Number of Entries Updated	ushort	16	Integer Number of Array entries modified by the command (i.e., number of cell structures to expect)
{1bytearray_cellstructure}+	N/A	{32}+	See Table 4.5.4.5.12

4.5.4.5.12 A661_1ByteArray_CellStructure

Table 4.5.4.5.12 – A661_1ByteArray_CellStructure

A661_ParameterStructure	Type	Size (bits)	Description
EntryIndex	ushort	16	Index to the Array item being modified by this cell.
ParameterValue	uchar	8	Value to be written to the Array item
UnusedPad	N/A	8	0

4.5.4.5.13 A661_ParameterStructure_2ByteArray

Table 4.5.4.5.13 – A661_ParameterStructure_2ByteArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
Number of Entries Updated	ushort	16	Integer Number of Array Entries modified by the command (i.e., number of cell structures to expect)
{2bytearray_cellstructure}+	N/A	{32}+	See Table 4.5.4.5.14

4.5.4.5.14 A661_2ByteArray_CellStructure

Table 4.5.4.5.14 – A661_2ByteArray_CellStructure

A661_ParameterStructure	Type	Size (bits)	Description
EntryIndex	ushort	16	Index to the Array item being modified by this cell.
ParameterValue	ushort/ short	16	The value to be written to the Array item

4.5.4.5.15 A661_ParameterStructure_EntryPopUpArray

Table 4.5.4.5.15 – A661_ParameterStructure_EntryPopUpArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_ENTRY_POP_UP_ARRAY
Number of Entries Updated	ushort	16	Integer Number of Popup Array Entries modified by the command (i.e., number of structures to expect)
{EntryPopUp_Structure}+	N/A	{32}+	See Table 4.5.4.5.16

4.5.4.5.16 A661_EntryPopUp_Structure

Table 4.5.4.5.16 – A661_EntryPopUp_Structure

A661_ParameterStructure	Type	Size (bits)	Description
EntryIndex	uchar	8	Index into the EntryPopUp array. Corresponds to the EnableArray, StringArray and/or Picture Array entry tuple being modified.
Enable	uchar	8	Value to be written to the EnableArray entry. A661_FALSE A661_TRUE
Picture	ushort	16	Picture reference, the value to be written to the Picture Array item
StringLength	ushort	16	Integer Size of the String for this EntryPopUp array entry. Size is given in bytes, including terminating NULL.
String	string	{32}+	List of char. String to be written to the StringArray entry. Followed by zero, one, two, or three NULL character(s) to be 32 bits aligned.

Note: This structure is an exception to the normal padding rules.

4.5.4.5.17 A661_ParameterStructure_EntryListArray

Table 4.5.4.5.17 – A661_ParameterStructure_EntryListArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	A661_ENTRY_ARRAY
Number of Entries Updated	ushort	16	Integer. Number of EntryList Array entries modified by the command (i.e., number of structures to expect)
{EntryList_Structure}+	N/A	{32}+	See Table 4.5.4.5.18

4.5.4.5.18 A661_EntryList_Structure

Table 4.5.4.5.18 – A661_EntryList_Structure

A661_ParameterStructure	Type	Size (bits)	Description
StringLength	ushort	16	Integer. Size of the Label String for this EntryList Array entry. Size is given in bytes, including terminating NULL.
EntryIndex	ushort	16	Index into the EntryList Array. Corresponds to the LabelStringArray and EnableArray entry pair being modified.
Enable	uchar	8	Value to be written to the EnableArray entry. A661_FALSE A661_TRUE
UnusedPad	N/A	8	0
String	string	8 * string length + PAD	List of char. String to be written to the LabelStringArray entry. Followed by zero, one, two, or three NULL character(s) to be 32 bits aligned.

4.5.4.5.19 A661_ParameterStructure_8ByteArray

Table 4.5.4.5.19 – A661_ParameterStructure_8ByteArray

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
Number of Entries Updated	ushort	16	Integer Number of Array Entries modified by the command (i.e., number of cell structures to expect)
{8bytearray_cellstructure}+	N/A	{96}+	See Table 4.5.4.5.20

4.5.4.5.20 A661_ParameterStructure_8ByteArray_CellStructure

Table 4.5.4.5.20 – A661_8ByteArray_CellStructure

A661_ParameterStructure	Type	Size (bits)	Description
EntryIndex	ushort	16	Index to the Array item being modified by this cell
UnusedPad	N/A	16	0
ParameterValue1	ulong/ long/float	32	The value to be written to the Array item
ParameterValue2	ulong/ long/float	32	The value to be written to the Array item

4.5.4.5.21 A661_ParameterStructure_BlockedBufferOfItems

For usage with MapHorz_ItemList widget, see description in Section 3.3.22.2.3. For usage with MapVert_ItemList widget, see description in Section 3.4.5.2.3.

4.5.4.5.22 A661_ParameterStructure_BufferOfExclusionItems

For usage with ExcludedRegionsExtension, see description in Section 8.5.7.

4.5.4.5.23 A661_ParameterStructure_App_Data

A661_ParameterStructure_App_Data is defined in Table 4.5.4.5.23.

Table 4.5.4.5.23 – A661_ParameterStructure_App_Data

A661_ParameterStructure	Type	Size (bits)	Description
ParameterIdent	ushort	16	Identifier of the parameter type
Size	ushort	16	Size of data in number of 32-bit words
Data	N/A	32 * Size	

4.5.4.5.24 A661_ParameterStructure BoundaryArray

See MapBoundary Widget definition in Section 3.9.15.

4.6 ARINC 661 Keyword Values

The following tables define numeric values associated with the ARINC 661 keywords:

Table 4.6-1 – Constant – Definition File (16 bits)

ARINC 661 Constant – Definition File (16 bits)	
A661_DF_MAGIC_NUMBER	0xA661

Table 4.6-2 – Block Codes (8 bits)

ARINC 661 Block Codes (8 bits)	
Specification range	*0x80 – 0xFF*
A661_BEGIN_BLOCK	0xB0
A661_BEGIN_CHARSET_ENCODING_BLOCK	0xC8
A661_BEGIN_LAYER_BLOCK	0xA0
A661_BEGIN_PICTURE	0xBC
A661_BEGIN_PICTURE_BLOCK	0xBB
A661_BEGIN_SYMBOL_BLOCK	0xF0
A661_BEGIN_ANIMATION_LAW_BLOCK	0xCA
A661_DF_FOOTER	0xE0
A661_END_BLOCK	0xD0
A661_END_CHARSET_ENCODING_BLOCK	0xC9
A661_END_LAYER_BLOCK	0xC0
A661_END_PICTURE	0xBD
A661_END_PICTURE_BLOCK	0xBE
A661_END_SYMBOL_BLOCK	0xF8
A661_END_ANIMATION_LAW_BLOCK	0xCB
Reserved for OEM Customization	*0x00 – 0x7F*

Table 4.6-3a – Commands

ARINC 661 Commands	
Specification range	*0xCA00 – 0xCAFF*
A661_CMD_CREATE	0xCA01
A661_CMD_CREATE_SYMBOL	0xCA04
A661_CMD_EXTEND	0xCA05
A661_CMD_CREATE_ANIMATION_LAW	0xCA06
A661_CMD_SET_PARAMETER	0xCA02
A661_CMD_UA_REQUEST	0xCA03
Reserved for OEM Customization	*0xCB00 – 0xCBFF*

Table 4.6-3b – Symbol Definition Commands

ARINC 661 Symbol Definition Commands (16 bits)	
Specification range	*0x9000 – 0x91FF*
A661_SYMBOL_DEFN_ARC_CIRCLE	0x9080
A661_SYMBOL_DEFN_ARC_ELLIPSE	0x9070
A661_SYMBOL_DEFN_BOUNDING_CIRCLE	0x9150
A661_SYMBOL_DEFN_BOUNDING_RECT	0x9160
A661_SYMBOL_DEFN_CIRCULAR	0x9130
A661_SYMBOL_DEFN_CROWN	0x9090
A661_SYMBOL_DEFN_END_GROUP	0x9032
A661_SYMBOL_DEFN_FOCUS	0x9000
A661_SYMBOL_DEFN_HIGHLIGHT	0x9010
A661_SYMBOL_DEFN_LEGEND_ANCHOR	0x9060
A661_SYMBOL_DEFN_LINE	0x90A0
A661_SYMBOL_DEFN_LINE_POLAR	0x90B0
A661_SYMBOL_DEFN_POLYLINE	0x90C0
A661_SYMBOL_DEFN_RECTANGLE	0x90D0
A661_SYMBOL_DEFN_RECTANGULAR	0x9120
A661_SYMBOL_DEFN_SET_COLOR	0x9020
A661_SYMBOL_DEFN_SET_FONT	0x9040
A661_SYMBOL_DEFN_SET_HALO	0x9050
A661_SYMBOL_DEFN_SET_LINE_STYLE	0x9030
A661_SYMBOL_DEFN_SET_ROTATION	0x9034
A661_SYMBOL_DEFN_SET_SCALE	0x903A
A661_SYMBOL_DEFN_SYMBOL_REF	0x9024
A661_SYMBOL_DEFN_SET_TRANSLATION	0x9038
A661_SYMBOL_DEFN_SIZE	0x9140
A661_SYMBOL_DEFN_START_GROUP	0x9036
A661_SYMBOL_DEFN_TEXT	0x9110
A661_SYMBOL_DEFN_TRIANGLE	0x90E0
A661_SYMBOL_DEFN_TRIANGLE_FAN	0x90F0
A661_SYMBOL_DEFN_TRIANGLE_STRIP	0x9100
Reserved for OEM Customization	*0x9200 – 0x9FFF*

Table 4.6-3c – Animation Law Definition Commands (16 bits)

ARINC 661 Animation Law Definition Commands	
Specification range	*0xD900 – 0xD97F*
A661_ANIMATION_LAW_DEFN_EASING_CURVE	0xD900
Reserved for OEM Customization	*0xD980 - 0xD9FF*

Table 4.6-4 – Notifications

ARINC 661 Notifications	
Specification range	*0xCC00 – 0xCCFF*
A661_NOTIFY_EXCEPTION	0xCC03
A661_NOTIFY_LAYER_EVENT	0xCC02
A661_NOTIFY_WIDGET_EVENT	0xCC01
Reserved for OEM Customization	*0xCD00 – 0xCFFF*

Table 4.6-5a – Notes

ARINC 661 Notifications	
Specification range	*0xDC00 – 0xDCFF*
A661_NOTE_CYCLIC_ACTIVATION	0xDC82
A661_NOTE_LAYER_IS_ACTIVE	0xDC02
A661_NOTE_LAYER_IS_INACTIVE	0xDC03
A661_NOTE_REINIT_LAYER_DATA	0xDC01
A661_NOTE_CONNECTOR_DATA	0xDC86
Reserved for OEM Customization	*0xDD00 – 0xDFFF*

Table 4.6-5b – Requests

ARINC 661 Requests	
Specification range	*0xDA00 – 0xDAFF*
A661_REQ_CURSOR_ON_WIDGET	0xDA05
A661_REQ_FOCUS_ON_WIDGET	0xDA03
A661_REQ_LAYER_ACTIVE	0xDA01
A661_REQ_LAYER_INACTIVE	0xDA02
A661_REQ_LAYER_VISIBLE	0xDA04
Reserved for OEM Customization	*0xDB00 – 0xDBFF*

Table 4.6-6 – ExceptionType

A661_ExceptionType	
Specification range	*0xF000 – 0xF0FF*
A661_ERR_BAD_COMMAND	0xF001
A661_ERR_CREATE_ABORTED	0xF002
A661_ERR_EXTEND_ABORTED	0xF008
A661_ERR_MEMORY_OVERLOAD	0xF005
A661_ERR_PROCESS_OVERLOAD	0xF006
A661_ERR_RENDERING_OVERLOAD	0xF007
A661_ERR_SET_ABORTED	0xF003
A661_ERR_UA_REQUEST_ABORTED	0xF004
Reserved for OEM Customization	*0xF100 – 0xFFFF*

Table 4.6-7 – Widgets (16 bits)

ARINC 661 Widgets (16 bits)	
Specification range	*0xA000 – 0xA7FF*
A661_ACTIVE_AREA	0xA010
A661_ANIMATION_GROUP	0xA1E3
A661_ANIMATION_ONPARAM	0xA1E4
A661_ANIMATION_ROTATION	0xA1E5
A661_ANIMATION_SCALE	0xA1E6
A661_ANIMATION_TRANSLATION	0xA1E7
A661_BASIC_CONTAINER	0xA020
A661_BLINKING_CONTAINER	0xA030
A661_BROADCAST_RECEIVER	0xA032
A661_BUFFER_FORMAT	0xA040
A661_CHECK_BUTTON	0xA050
A661_COMBO_BOX	0xA070
A661_COMBO_BOX_EDIT	0xA0E0
A661_CONNECTOR	0xA080
A661_CURSOR_OVER	0xA480
A661_CURSOR_POS_OVERLAY	0xA090
A661_CURSOR_REF	0xA470
A661_DATA_CONNECTOR	0xA088
A661_EDIT_BOX_MASKED	0xA0A0
A661_EDIT_BOX_MULTI_LINE	0xA0B0
A661_EDIT_BOX_NUMERIC	0xA0C0
A661_EDIT_BOX_NUMERIC_BCD	0xA0C2
A661_EDIT_BOX_TEXT	0xA0D0
A661_EVENT_HANDLER	0xA600
A661_EXTERNAL_SOURCE	0xA188
A661_FOCUS_IN	0xA498
A661_FOCUS_LINK	0xA490
A661_FOCUS_OUT	0xA499
A661_FRAMING_REFRESH_CONTAINER	0xA380
A661_GESTURE_AREA	0xA501
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_POLYLINE	0xA138
A661_GP_RECTANGLE	0xA140
A661_GP_TRIANGLE	0xA150
A661_GP_TRIANGLE_FAN	**0xA151**
A661_GP_TRIANGLE_STRIP	**0xA152**
A661_INK_AREA	0xA502
A661_KEYBOARD_AREA	0xA1E1
A661_SCROLL_WHEEL_AREA	0xA1E2
A661_LABEL	0xA160
A661_LABEL_COMPLEX	0xA170
A661_MAP_BOUNDARY	0xA1BA
A661_MAP_EXTERNAL_SOURCE	**0xA224**

ARINC 661 Widgets (16 bits)	
A661_MAP_GRID	0xA178
A661_MAPHORZ	0xA1B0
A661_MAPHORZ_CONTAINER	0xA1B8
A661_MAPHORZ_ITEMLIST	0xA180
A661_MAPHORZ_PANEL	0xA189
A661_MAPHORZ_SOURCE	0xA1A0
A661_MAPHORZ_VERTEXBUFFER	0xA1A9
A661_MAPVERT	0xA1B2
A661_MAPVERT_ITEMLIST	0xA1B3
A661_MAPVERT_SOURCE	0xA1B4
A661_MAPVERT_CONTAINER	0xA1B5
A661_MAPVERT_PANEL	0xA1B6
A661_MASK_CONTAINER	0xA1C0
A661_MENU_BAR	0xA1D0
A661_MULTI_STATE_BUTTON	0xA348
A661_MUTUALLY_EXCLUSIVE_CONTAINER	0xA400
A661_NO_SERVICE_MONITOR	0xA4F0
A661_NUMERIC_READOUT	0xA406
A661_PAGING_CONTAINER	0xA408
A661_PANEL	0xA1F0
A661_PICTURE	0xA200
A661_PICTURE_ANIMATED	0xA210
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_POP_UP_PANEL_BUTTON	0xA4E0
A661_PROXY_BUTTON	0xA420
A661_PUSH_BUTTON	0xA2A0
A661_RADIO_BOX	0xA2B0
A661_ROTATION_CONTAINER	0xA2D0
A661_DATA_SCALING_LONG	0xA2D8
A661_DATA_SCALING_ULONG	0xA2D9
A661_DATA_SCALING_FR180	0xA2E0
A661_SCALE_CONTAINER	**0xA2F2**
A661_SCROLL_LIST	0xA2F0
A661_SCROLL_PANEL	0xA300
A661_SELECTION_LIST_BUTTON	0xA370
A661_SHUFFLE_TO_FIT_CONTAINER	0xA4B0
A661_SIZE_TO_FIT_CONTAINER	0xA4A0
A661_SLIDER	0xA440
A661_SYMBOL	0xA310
A661_SYMBOL_ANIMATED	0xA450
A661_SYMBOL_PUSH_BUTTON	0xA4C0
A661_SYMBOL_TOGGLE_BUTTON	0xA4D0
A661_TABBED_PANEL	0xA320
A661_TABBED_PANEL_GROUP	0xA330
A661_TOGGLE_BUTTON	0xA340
A661_TOUCH_AREA	0xA500
A661_TRANSLATION_CONTAINER	0xA360

ARINC 661 Widgets (16 bits)	
A661_TREE	0xA222
A661_WATCHDOG_CONTAINER	0xA460
Reserved	*0xA060*
Reserved	*0xA190*
Reserved	*0xA1E0*
Reserved	*0xA260*
Reserved	*0xA2C0*
Reserved	*0xA350*
Reserved	*0xA661*
Reserved for OEM Customization	*0xA800 to 0xAFFF*

Table 4.6-8 – Parameter Types

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
Specification range	*0xB000 – 0xB7FF*	
A661_AC_LAT	0xB010	A661_ParameterStructure_4Bytes
A661_AC_LAT_LONG	0xB030	A661_ParameterStructure_8Bytes
A661_AC_LONG	0xB020	A661_ParameterStructure_4Bytes
A661_AC_ORIENTATION	0xB040	A661_ParameterStructure_4Bytes
A661_ACTIVE_TABBED_PANEL	0xB050	A661_ParameterStructure_2Bytes
A661_ALIGNMENT	0xB49A	A661_ParameterStructure_1Byte
A661_ALLOW_CHANGE	0xB6A1	A661_ParameterStructure_1Byte
A661_ALPHA_MASK	0xB060	A661_ParameterStructure_4Bytes
A661_ALTERNATE_FLAG	0xB088	A661_ParameterStructure_1Byte
A661_ALTERN_PICTURE_REFERENCE	0xB080	A661_ParameterStructure_2Bytes
A661_ALTERN_SYMBOL_REFERENCE	0xB4C1	A661_ParameterStructure_2Bytes
A661_ANIMATION_DELAY	0xB702	A661_ParameterStructure_2Bytes
A661_ANIMATION_DURATION	0xB703	A661_ParameterStructure_2Bytes
A661_ANIMATION_FLAG	0xB090	A661_ParameterStructure_1Byte
A661_ANIMATION_LAW_REF	0xB704	A661_ParameterStructure_2Bytes
A661_ANIMATION_REPETITION	0xB705	A661_ParameterStructure_2Bytes
A661_ANIMATION_STATE	0xB706	A661_ParameterStructure_1Byte
A661_ANIMATION_TYPE	0xB098	A661_ParameterStructure_1Byte
A661_APP_DATA	0xB6D0	A661_ParameterStructure_App_Data
A661_AUTO_CLOSURE	0xB1C2	A661_ParameterStructure_1Byte
A661_AUTO_FOCUS_MOTION	0xB2B9	A661_ParameterStructure_1Byte
A661_BLINKING_TYPE	0xB0A8	A661_ParameterStructure_1Byte
A661_BLOCKED_BUFFER_OF_MAPITEM	0xB121	A661_ParameterStructure_BlockedBufferOfItems Refer to Table 3.3.22.2.3
A661_BLOCKED_BUFFER_OF_MAPVERT_ITEMS	0xB126	A661_ParameterStructure_BlockedBufferOfItems (Vert) Refer to Table 3.4.5.2.3.
A661_BOUND_SIZE_X	0xB0E0	A661_ParameterStructure_4Bytes
A661_BOUND_SIZE_XY	0xB100	A661_ParameterStructure_8Bytes
A661_BOUND_SIZE_Y	0xB0F0	A661_ParameterStructure_4Bytes
A661_BOUND_X	0xB0B0	A661_ParameterStructure_4Bytes
A661_BOUND_XY	0xB0D0	A661_ParameterStructure_8Bytes
A661_BOUND_Y	0xB0C0	A661_ParameterStructure_4Bytes
A661_BOUNDARY_ARRAY	0xB6C1	A661_Parameter_Structure_BoundaryArray
A661_BOUNDARY_CHECK_MODE	0xB122	A661_ParameterStructure_1Byte
A661_BROADCAST_DATA	0xB0C2	Depends of the structure of parameter to broadcast
A661_BUFFER_OF_FILL_STYLES	0xB0F8	A661_ParameterStructure_BufferOfFillStyles Refer to Table 3.4.1.1-1

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
A661_BUFFER_OF_MAPITEM	0xB120	A661_ParameterStructure_BufferOfItems Refer to Table 3.3.22.2.2
A661_BUFFER_OF_MAPVERT_ITEMS	0xB125	A661_ParameterStructure_BufferOfItems (Vert) Refer to Table 3.4.5.2.2
A661_BUFFER_OF_PARAM	0xB110	A661_ParameterStructure_Buffer
A661_BUTTON_POS	**0xB641**	**A661_ParameterStructure_4Bytes**
A661_BUTTON_SIZE	**0xB642**	**A661_ParameterStructure_4Bytes**
A661_CACHING_ENABLE	0xB117	A661_ParameterStructure_1Byte
A661_CENTER_X	0xB130	A661_ParameterStructure_4Bytes
A661_CENTER_XY	0xB150	A661_ParameterStructure_8Bytes
A661_CENTER_Y	0xB140	A661_ParameterStructure_4Bytes
A661_CLAMPING	0xB501	A661_ParameterStructure_1Byte
A661_CLEAR	0xB000	A661_ParameterStructure_1Byte
A661_CLOSED	**0xB1E2**	**A661_ParameterStructure_1Byte**
A661_COARSE_PAGE_DELTA	**0xB6E3**	**A661_ParameterStructure_2Bytes**
A661_COLOR_INDEX	0xB160	A661_ParameterStructure_1Byte
A661_CRP_LAT	0xB162	A661_ParameterStructure_4Bytes
A661_CRP_LAT_LONG	0xB166	A661_ParameterStructure_8Bytes
A661_CRP_LONG	0xB164	A661_ParameterStructure_4Bytes
A661_CURSOR_INDEX	0xB6A0	A661_ParameterStructure_1Byte
A661_CURSOR_POS	0xB170	A661_ParameterStructure_2Bytes
A661_CURSOR_POS_BYTE	0xB172	A661_ParameterStructure_1Byte
A661_DIRECTION	**0xB713**	**A661_ParameterStructure_1Byte**
A661_EFFECTIVITY_ARRAY	0xB6C8	A661_ParameterStructure_1ByteArray
A661_ENABLE	0xB180	A661_ParameterStructure_1Byte
A661_ENABLE_ARRAY (deprecated in Supplement 4)	0xB1A0	A661_ParameterStructure_EnableArray
A661_ENABLE_ENTRY_ARRAY	0xB1A8	A661_ParameterStructure_1ByteArray
A661_ENABLE_HANDLER	0x**B1A1**	A661_ParameterStructure_1Byte
A661_END_ANGLE	0xB1B0	A661_ParameterStructure_4Bytes
A661_END_SCALED_VALUE	0xB505	A661_ParameterStructure_4Bytes
A661_END_VALUE	0xB503	A661_ParameterStructure_4Bytes
A661_ENTRY_ARRAY	0xB190	A661_ParameterStructure_EntryListArray
A661_ENTRY_POP_UP_ARRAY	0xB1C0	A661_ParameterStructure_EntryPopUpArray
A661_ENTRY_VALID	0xB570	A661_ParameterStructure_1Byte
A661_EVENT_FLAG	0xB1D0	A661_ParameterStructure_1Byte
A661_EXCLUDED_REGIONS_LIST	0xB6C9	A661_ParameterStructure_BufferOfExclusionItems
A661_FILLED	**0xB1E1**	**A661_ParameterStructure_1Byte**
A661_FILL_INDEX	0xB1E0	A661_ParameterStructure_1Byte
A661_FINE_PAGE_DELTA	**0xB6E2**	**A661_ParameterStructure_2Bytes**
A661_FIRST_ACCESS_ENTRY	0xB1F0	A661_ParameterStructure_2Bytes
A661_FIRST_ACCESS_UA_ENTRY	0xB1F8	A661_ParameterStructure_2Bytes
A661_FIRST_VISIBLE_ENTRY	0xB200	A661_ParameterStructure_2Bytes
A661_FONT	0xB590	A661_ParameterStructure_1Byte
A661_FORMAT_STRING	0xB550	A661_ParameterStructure_String
A661_FRAME_X	0xB210	A661_ParameterStructure_4Bytes
A661_FRAME_XY	0xB230	A661_ParameterStructure_8Bytes
A661_FRAME_Y	0xB220	A661_ParameterStructure_4Bytes
A661_FROM_ANGLE	0xB707	A661_ParameterStructure_4Bytes
A661_FROM_SCALE_X	0xB708	A661_ParameterStructure_4Bytes
A661_FROM_SCALE_XY	0xB715	A661_ParameterStructure_8Bytes
A661_FROM_SCALE_Y	0xB709	A661_ParameterStructure_4Bytes
A661_FROM_VALUE	0xB70A	A661_ParameterStructure_4Bytes
A661_FROM_VALUE_X	0xB70B	A661_ParameterStructure_4Bytes
A661_FROM_VALUE_XY	0xB717	A661_ParameterStructure_8Bytes

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
A661_FROM_VALUE_Y	0xB70C	A661_ParameterStructure_4Bytes
A661_HALO	**0xB1E8**	**A661_ParameterStructure_1Byte**
A661_HOME_X	**0xB219**	**A661_ParameterStructure_4Bytes**
A661_HOME_Y	**0xB21A**	**A661_ParameterStructure_4Bytes**
A661_HORIZONTAL_SCROLL	**0xB217**	**A661_ParameterStructure_1Byte**
A661_INDEX_OF_FREQUENCY	**0xB091**	**A661_ParameterStructure_1Byte**
A661_INITIAL_FOCUS	0xB6B0	A661_ParameterStructure_1Byte
A661_INNER_RADIUS	0xB240	A661_ParameterStructure_4Bytes
A661_INNER_STATE_CHECK	0xB244	A661_ParameterStructure_1Byte
A661_INNER_STATE_TOGGLE	0xB258	A661_ParameterStructure_1Byte
A661_INSET_SIZE	**0xB051**	**A661_ParameterStructure_4Bytes**
A661_ITEM_SPACING	**0xB5C1**	**A661_ParameterStructure_4Bytes**
A661_KEY_ARRAY	0xB700	A661_ParameterStructure_2ByteArray
A661_LAYOUT	**0xB5F0**	**A661_ParameterStructure_1Byte**
A661_LEGEND_ALIGNMENT	**0xB49B**	**A661_ParameterStructure_1Byte**
A661_LEGEND_AREA_SIZE_X	**0xB451**	**A661_ParameterStructure_4Bytes**
A661_LEGEND_POSITION	0xB260	A661_ParameterStructure_1Byte
A661_LEGEND_REMOVED	**0xB452**	**A661_ParameterStructure_1Byte**
A661_LINE_DELTA_X	**0xB211**	**A661_ParameterStructure_4Bytes**
A661_LINE_DELTA_Y	**0xB212**	**A661_ParameterStructure_4Bytes**
A661_LINE_LENGTH	0xB270	A661_ParameterStructure_4Bytes
A661_LSB_MULTIPLE	**0xB631**	**A661_ParameterStructure_2Bytes**
A661_MAJOR_TICK_INTERVAL	**0xB600**	**A661_ParameterStructure_4Bytes**
A661_MAJOR_TICK_REFERENCE	**0xB608**	**A661_ParameterStructure_4Bytes**
A661_MAX_VALUE	**0xB610**	**A661_ParameterStructure_4Bytes**
A661_MENU_HORIZONTAL	**0xB640**	**A661_ParameterStructure_1Byte**
A661_MIN_VALUE	**0xB620**	**A661_ParameterStructure_4Bytes**
A661_MINOR_TICK_MULTIPLE	**0xB630**	**A661_ParameterStructure_2Bytes**
A661_MAP_SYNCHRONIZATION_NUMBER	0xB277	A661_ParameterStructure_2Bytes
A661_MAPGRID_CELLSIZE	0xB274	A661_ParameterStructure_8Bytes
A661_MAPGRID_OFFSET	0xB278	A661_ParameterStructure_8Bytes
A661_MASK_ENABLED	0xB290	A661_ParameterStructure_1Byte
A661_MASK_REFERENCE	0xB280	A661_ParameterStructure_2Bytes
A661_MINMAX_VALUES	0xB580	A661_ParameterStructure_8Bytes
A661_MULTI_STATE_VALUE	0xB5A0	A661_ParameterStructure_1Byte
A661_NEGATIVE_STRING	**0xB4D4**	**A661_ParameterStructure_String**
A661_NEXT_FOCUS_IN_APP_ID	0xB2BA	A661_ParameterStructure_2Bytes
A661_NEXT_FOCUS_IN_LAYER_ID	0xB2BB	A661_ParameterStructure_1Byte
A661_NEXT_FOCUS_IN_ID	0xB2BC	A661_ParameterStructure_2Bytes
A661_NEXT_FOCUSED_WIDGET	0xB2B8	A661_ParameterStructure_2Bytes
A661_NUMBER_OF_ENTRIES	0xB2A0	A661_ParameterStructure_2Bytes
A661_NUMBER_OF_KEYS	0xB701	A661_ParameterStructure_1Byte
A661_NUMBER_OF_PICTURES	**0xB2A2**	**A661_ParameterStructure_2Bytes**
A661_NUMBER_OF_STATES	0xB5A1	A661_ParameterStructure_1Byte
A661_NUMBER_OF_SYMBOLS	**0xB2A4**	**A661_ParameterStructure_2Bytes**
A661_NUMBER_OF_TICS_COARSE	0xB0A2	A661_ParameterStructure_2Bytes
A661_NUMBER_OF_TICS_FINE	0xB0A4	A661_ParameterStructure_2Bytes
A661_NUMBER_OF_TOUCH_POINTS	0xB0A0	A661_ParameterStructure_1Byte
A661_NUMBER_OF_TURNS	0xB714	A661_ParameterStructure_1Byte
A661_NUMBER_OF_UA_ENTRIES	0xB2A8	A661_ParameterStructure_2Bytes
A661_NUMBER_OF_VISIBLE_CHILDREN	0xB5D0	A661_ParameterStructure_2Bytes
A661_NUMERIC_MASK	0xB2B0	A661_ParameterStructure_4Bytes
A661_OPENING_ENTRY	0xB560	A661_ParameterStructure_2Bytes
A661_OPENING_MODE	**0xB561**	**A661_ParameterStructure_1Byte**

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
A661_OPEN_ENTRY_ARRAY	0xB627	A661_ParameterStructure_1ByteArray
A661_ORIENTATION	0xB2C0	A661_ParameterStructure_4Bytes
A661_OUTER_RADIUS	0xB2D0	A661_ParameterStructure_4Bytes
A661_PAGE_CONTROL_POSITION	0xB6E1	A661_ParameterStructure_1Byte
A661_PAGE_DELTA_X	0xB213	A661_ParameterStructure_4Bytes
A661_PAGE_DELTA_Y	0xB214	A661_ParameterStructure_4Bytes
A661_PAGE_WRAP_TYPE	0xB6E0	A661_ParameterStructure_1Byte
A661_PARAM_BUFFER_INDEX	0xB111	A661_ParameterStructure_2Bytes
A661_PARENT_ENTRY_ARRAY	0xB628	A661_ParameterStructure_1ByteArray
A661_PICTURE_ARRAY	0xB2E0	A661_ParameterStructure_PictureArray Refer to Table 3.3.31.1-1
A661_PICTURE_ENTRY_ARRAY	0xB2E8	A661_ParameterStructure_2ByteArray
A661_PICTURE_POSITION	0xB2F1	A661_ParameterStructure_1Byte
A661_PICTURE_REFERENCE	0xB2F0	A661_ParameterStructure_2Bytes
A661_POPUP_POS_X	0xB1C3	A661_ParameterStructure_4Bytes
A661_POPUP_POS_Y	0xB1C4	A661_ParameterStructure_4Bytes
A661_POPUP_SIZE_X	0xB1C5	A661_ParameterStructure_4Bytes
A661_POPUP_SIZE_Y	0xB1C6	A661_ParameterStructure_4Bytes
A661_POSITIVE_STRING	0xB4D3	A661_ParameterStructure_String
A661_POS_X	0xB300	A661_ParameterStructure_4Bytes
A661_POS_X2	0xB330	A661_ParameterStructure_4Bytes
A661_POS_X3	0xB360	A661_ParameterStructure_4Bytes
A661_POS_XY	0xB320	A661_ParameterStructure_8Bytes
A661_POS_XY2	0xB350	A661_ParameterStructure_8Bytes
A661_POS_XY3	0xB380	A661_ParameterStructure_8Bytes
A661_POS_Y	0xB310	A661_ParameterStructure_4Bytes
A661_POS_Y2	0xB340	A661_ParameterStructure_4Bytes
A661_POS_Y3	0xB370	A661_ParameterStructure_4Bytes
A661_PRP_LAT	0xB390	A661_ParameterStructure_4Bytes
A661_PRP_LAT_LONG	0xB3B0	A661_ParameterStructure_8Bytes
A661_PRP_LONG	0xB3A0	A661_ParameterStructure_4Bytes
A661_PRP_SCREEN_X	0xB3C0	A661_ParameterStructure_4Bytes
A661_PRP_SCREEN_XY	0xB3E0	A661_ParameterStructure_8Bytes
A661_PRP_SCREEN_Y	0xB3D0	A661_ParameterStructure_4Bytes
A661_PRP_X	0xB3B2	A661_ParameterStructure_4Bytes
A661_PRP_XY	0xB3B4	A661_ParameterStructure_8Bytes
A661_PRP_Y	0xB3B3	A661_ParameterStructure_4Bytes
A661_RADIUS	0xB3F0	A661_ParameterStructure_4Bytes
A661_RANGE	0xB400	A661_ParameterStructure_4Bytes
A661_RANGE_X	0xB402	A661_ParameterStructure_4Bytes
A661_RANGE_XY	0xB404	A661_ParameterStructure_8Bytes
A661_RANGE_Y	0xB403	A661_ParameterStructure_4Bytes
A661_REF_POS_X	0xB690	A661_ParameterStructure_4Bytes
A661_REF_POS_XY	0xB691	A661_ParameterStructure_8Bytes
A661_REF_POS_Y	0xB692	A661_ParameterStructure_4Bytes
A661_REFRESH	0xB670	A661_ParameterStructure_1Byte
A661_REORDER	0xB0AB	A661_ParameterStructure_1Byte
A661_ROTATION_ANGLE	0xB410	A661_ParameterStructure_4Bytes
A661_SCALE_X	0xB0AC	A661_ParameterStructure_4Bytes
A661_SCALE_XY	0xB0AE	A661_ParameterStructure_8Bytes
A661_SCALE_Y	0xB0AD	A661_ParameterStructure_4Bytes
A661_SCREEN_RANGE	0xB420	A661_ParameterStructure_4Bytes
A661_SELECT_ALL	0xB438	A661_ParameterStructure_1Byte
A661_SELECTED_ENTRY	0xB430	A661_ParameterStructure_2Bytes
A661_SELECTED_ENTRY_ARRAY	0xB439	A661_ParameterStructure_1ByteArray

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
A661_SELECTING_HEIGHT	**0xB431**	**A661_ParameterStructure_4Bytes**
A661_SELECTING_WIDTH	**0xB432**	**A661_ParameterStructure_4Bytes**
A661_SHEET_SIZE_X	**0xB215**	**A661_ParameterStructure_4Bytes**
A661_SHEET_SIZE_Y	**0xB216**	**A661_ParameterStructure_4Bytes**
A661_SHIFT_FIRST_VISIBLE_ENTRY	0xB440	A661_ParameterStructure_2Bytes
A661_SHOW_FAIL	0xB671	A661_ParameterStructure_1Byte
A661_SHOW_MAJOR_TICKS	**0xB601**	**A661_ParameterStructure_1Byte**
A661_SHUFFLE_TO_FIT_MODE	0xB5E0	A661_ParameterStructure_1Byte
A661_SIZE_TO_FIT_MODE	0xB5C0	A661_ParameterStructure_1Byte
A661_SIZE_X	0xB450	A661_ParameterStructure_4Bytes
A661_SIZE_XY	0xB470	A661_ParameterStructure_8Bytes
A661_SIZE_Y	0xB460	A661_ParameterStructure_4Bytes
A661_SOURCE_DX	0xB662	A661_ParameterStructure_4Bytes
A661_SOURCE_DXDY	0xB665	A661_ParameterStructure_8Bytes
A661_SOURCE_DY	0xB663	A661_ParameterStructure_4Bytes
A661_SOURCE_REF	**0xB666**	**A661_ParameterStructure_2Bytes**
A661_SOURCE_X	0xB660	A661_ParameterStructure_4Bytes
A661_SOURCE_XY	0xB664	A661_ParameterStructure_8Bytes
A661_SOURCE_Y	0xB661	A661_ParameterStructure_4Bytes
A661_START_ANGLE	0xB480	A661_ParameterStructure_4Bytes
A661_START_SCALED_VALUE	0xB504	A661_ParameterStructure_4Bytes
A661_START_VALUE	0xB502	A661_ParameterStructure_4Bytes
A661_STOP_BACKWARD	0xB6B2	A661_ParameterStructure_1Byte
A661_STOP_FORWARD	0xB6B1	A661_ParameterStructure_1Byte
A661_STRING	0xB490	A661_ParameterStructure_String
A661_STRING_ALTERNATE	0xB498	A661_ParameterStructure_String
A661_STRING_ARRAY	0xB4A0	A661_ParameterStructure_StringArray
A661_STYLE_SET	0xB4B0	A661_ParameterStructure_2Bytes
A661_STYLE_SET_ARRAY	0xB4B1	A661_ParameterStructure_2ByteArray
A661_STYLE_SET_ARRAY_SIZE	0xB4B2	A661_ParameterStructure_1Byte
A661_SYMBOL_ENTRY_ARRAY	0xB2E9	A661_ParameterStructure_2ByteArray
A661_SYMBOL_POSITION	**0xB4C2**	**A661_ParameterStructure_1Byte**
A661_SYMBOL_REFERENCE	0xB4C0	A661_ParameterStructure_2Bytes
A661_SYNC_FRAME	0xB4C7	A661_ParameterStructure_2Bytes
A661_TAB_DIRECTION_WIDGET_ID_ARRAY	0xB672	A661_ParameterStructure_2ByteArray
A661_TAB_POSITION	**0xB052**	**A661_ParameterStructure_1Byte**
A661_TARGET_WIDGET_ID	0xB650	A661_ParameterStructure_2Bytes
A661_TICS_COARSE	0xB4D0	A661_ParameterStructure_4Bytes
A661_TICS_COARSE_8BYTE	0xB4D2	A661_ParameterStructure_8Bytes
A661_TICS_COARSE_ARRAY	0xB0A6	A661_ParameterStructure_8ByteArray
A661_TICS_FINE	0xB4E0	A661_ParameterStructure_4Bytes
A661_TICS_FINE_8BYTE	0xB4E2	A661_ParameterStructure_8Bytes
A661_TICS_FINE_ARRAY	0xB0AA	A661_ParameterStructure_8ByteArray
A661_TO_ANGLE	0xB70D	A661_ParameterStructure_4Bytes
A661_TO_SCALE_X	0xB70E	A661_ParameterStructure_4Bytes
A661_TO_SCALE_XY	0xB716	A661_ParameterStructure_8Bytes
A661_TO_SCALE_Y	0xB70F	A661_ParameterStructure_4Bytes
A661_TO_VALUE	0xB710	A661_ParameterStructure_4Bytes
A661_TO_VALUE_X	0xB711	A661_ParameterStructure_4Bytes
A661_TO_VALUE_XY	0xB718	A661_ParameterStructure_8Bytes
A661_TO_VALUE_Y	0xB712	A661_ParameterStructure_4Bytes
A661_TRANSLATION_X	0xB4F0	A661_ParameterStructure_4Bytes
A661_TRANSLATION_XY	0xB510	A661_ParameterStructure_8Bytes
A661_TRANSLATION_Y	0xB500	A661_ParameterStructure_4Bytes

ARINC 661 Parameter Types (16 bits)ParameterIdent used in the ParameterStructure		Type of Structure Used (Refer to Section 4.5.4.5)
A661_TREE_NODE_STYLE_ARRAY	**0xB629**	**A661_ParameterStructure_2ByteArray**
A661_UA_POSITION_FLAG	**0xB1C1**	**A661_ParameterStructure_1Byte**
A661_UNSELECT_ALL	0xB43A	A661_ParameterStructure_1Byte
A661_VALUE	0xB520	A661_ParameterStructure_4Bytes
A661_VALUE_8BYTE	0xB522	A661_ParameterStructure_8Bytes
A661_VERTICES_ARRAY	0xB528	A661_ParameterStructure_8ByteArray
A661_VERTICAL_SCROLL	**0xB218**	**A661_ParameterStructure_1Byte**
A661_VISIBLE	0xB530	A661_ParameterStructure_1Byte
A661_VISIBLE_CHILD	0xB540	A661_ParameterStructure_2Bytes
A661_VISIBLE_CHILD_INDEX	0xB680	A661_ParameterStructure_2Bytes
A661_WORD_WRAP	**0xB4A8**	**A661_ParameterStructure_1Byte**
A661_WRAP_STYLE	**0xB4A9**	**A661_ParameterStructure_1Byte**
Reserved for OEM Customization	*0xB800 to 0xBFFF*	

Table 4.6-9 – Event Types

ARINC 661 Event Types (16 bits)	
Specification range	*0xE000 – 0xE7FF*
A661_EVT_ANIMATION_STATUS_CHANGE	0xE509
A661_EVT_CURSOR_ENTER	0xE300
A661_EVT_CURSOR_ENTRY_EVENT	**0xE056**
A661_EVT_CURSOR_EVENT	0xE305
A661_EVT_CURSOR_EXIT	0xE320
A661_EVT_CURSOR_INSIDE	0xE310
A661_EVT_CURSOR_POS_CHANGE	0xE010
A661_EVT_EDITBOX_OPENED	0xE110
A661_EVT_FIRST_VIS_ENTRY_CHANGE	0xE020
A661_EVT_FRAME_POS_CHANGE	0xE030
A661_EVT_GESTURE_DIR_FLICK	0xE508
A661_EVT_GESTURE_DOUBLE_TAP	0xE502
A661_EVT_GESTURE_DRAG	0xE506
A661_EVT_GESTURE_FLICK	0xE507
A661_EVT_GESTURE_HOLD	**0xE50A**
A661_EVT_GESTURE_PINCH	**0xE504**
A661_EVT_GESTURE_PRESS_AND_HOLD	**0xE503**
A661_EVT_GESTURE_ROTATE	**0xE505**
A661_EVT_GESTURE_TAP	0xE501
A661_EVT_INCREMENT	0xE006
A661_EVT_ITEM_SYNCHRONIZATION	0xE150
A661_EVT_ITEM_IN_BOUNDS	0xE141
A661_EVT_ITEM_OUT_OF_BOUNDS	0xE140
A661_EVT_KEY	0xE005
A661_EVT_MAPHORZ_ORIENTATION	**0xE06E**
A661_EVT_MAPHORZ_PROJECTION_REFERENCE	**0xE06C**
A661_EVT_MAPHORZ_RANGE	**0xE06A**
A661_EVT_MAPVERT_PROJECTION_REFERENCE	**0xE06D**
A661_EVT_MAPVERT_RANGE_X	**0xE06B**
A661_EVT_MAPVERT_RANGE_Y	**0xE06F**
A661_EVT_MULT_SELECTION	0xE058
A661_EVT_OFFSCREEN	0xE0E0

ARINC 661 Event Types (16 bits)	
A661_EVT_ONSCREEN	0xE0E1
A661_EVT_POPUP_CLOSED	0xE040
A661_EVT_POPUP_PANEL_CLOSED	0xE120
A661_EVT_SEL_ENTRY_CHANGE	0xE050
A661_EVT_TREE_ENTRY_OPEN	**0xE052**
A661_EVT_TREE_ENTRY_CLOSE	**0xE054**
A661_EVT_SELECTION	0xE060
A661_EVT_SELECTION_MAP	0xE068
A661_EVT_STATE_CHANGE	0xE070
A661_EVT_STRING_CHANGE	0xE080
A661_EVT_STRING_CHANGE_ABORTED	0xE090
A661_EVT_STRING_CONFIRMED	0xE0A0
A661_EVT_TABBED_PANEL_CHANGE	0xE0B0
A661_EVT_TOUCH	0xE500
A661_EVT_VALUE_CHANGE	0xE0C0
A661_EVT_VISIBLE_CHILD	0xE0D0
A661_EVT_VISIBLE_CHILD_INDEX	0xE0D1
A661_EVT_WATCHDOG_EXPIRED	0xE200
A661_EVT_WATCHDOG_NORMAL	0xE210
Reserved for OEM Customization	*0xE800 to 0xEFFF*

Table 4.6-10 – Boolean Constant Values

ARINC 661 Boolean Constant Values	
A661_FALSE	0x00
A661_TRUE	0x01
A661_TRUE_WITH_VALIDATION	0x02

Table 4.6-11 – Integer Constant Values

ARINC 661 Integer Constant Values	
On 8 bits	
Specification range	*0x00 – 0xCF*
A661_ABORT	0x02
A661_ABORTED	0x02
A661_ABSENT	0x10
A661_BOTH_CLAMPED	0x01
A661_BOTTOM	0x12
A661_BOTTOM_CENTER	0x30
A661_BOTTOM_LEFT	0x31
A661_BOTTOM_RIGHT	0x32
A661_BOTTOM_TO_TOP	0x40
A661_BOUNDARY_CHECK	0xBC
A661_CDS_DEPENDENT	0x1E
A661_CENTER	0x15
A661_CLOCKWISE	0x00
A661_COMPLETED	0x01
A661_COUNTER_CLOCKWISE	0x01
A661_DEACTIVATE	**0x04**
A661_DONT_RUN	0x00
A661_DOWN	0x1A

ARINC 661 Integer Constant Values	
A661_DRAW_LINE_TO_CURSOR	0x36
A661_EDB_ALL_CHANGE	0x01
A661_EDB_CHANGE_CONFIRMED	0x00
A661_EDB_OPEN_CLOSE	0x02
A661_EDITING	0x1B
A661_END	**0x03**
A661_ERROR	0x1C
A661_EVENT_NONE	0x00
A661_EXCLUDED_RECTANGLE	0x01
A661_EXCLUDED_TRIANGLE	0x02
A661_EXCLUDED_CIRCLE	0x03
A661_FILLED_POLY_START	0x2B
A661_FILLED_POLY_START_INTERACTIVE	0xAB
A661_FROM	**0x05**
A661_GESTURE_ABORT	0x04
A661_GESTURE_CANDIDATE	**0x05**
A661_GESTURE_DOUBLE_TAP	0x02
A661_GESTURE_DRAG	0x06
A661_GESTURE_END	0x03
A661_GESTURE_FLICK	0x07
A661_GESTURE_FLICK_DOWN	0x0B
A661_GESTURE_FLICK_LEFT	0x08
A661_GESTURE_FLICK_RIGHT	0x09
A661_GESTURE_FLICK_UP	0x0A
A661_GESTURE_HOLD	**0x0C**
A661_GESTURE_PINCH	0x04
A661_GESTURE_PRESS_AND_HOLD	0x03
A661_GESTURE_ROTATE	0x05
A661_GESTURE_START	0x01
A661_GESTURE_TAP	0x01
A661_GESTURE_UPDATE	0x02
A661_HIGHLIGHT_ONLY	0x04
A661_HYPHENATE	**0x02**
A661_IN_BOUNDS	0x01
A661_ITEM_STYLE	0x20
A661_ITEM_SYNCHRONIZATION	0x51
A661_LEFT	0x13
A661_LEFT_TO_RIGHT	0x41
A661_LEGEND	0x21
A661_LEGEND_ANCHOR	0x22
A661_LEGEND_ANCHOR_ROTATED	0x35
A661_LEGEND_AND_HIGHLIGHT	0x05
A661_LEGEND_AND_POPUP	0x03
A661_LEGEND_AND_POPUP_AND_HIGHLIGHT	0x07
A661_LEGEND_COMBO	0x38
A661_LEGEND_HIGHLIGHT	0x58
A661_LEGEND_ONLY	0x01
A661_LEGEND_POP_UP	0x23
A661_LEGEND_READOUT	**0x45**
A661_LEGEND_READOUT_COMBO	**0x48**
A661_LEGEND_READOUT_FORMAT_STRING	**0x44**

ARINC 661 Integer Constant Values	
A661_LINE_ARC	0x24
A661_LINE_ARC_INTERACTIVE	0xA4
A661_LINE_SEGMENT	0x25
A661_LINE_SEGMENT_INTERACTIVE	0xA5
A661_LINE_START	0x26
A661_LINE_START_INTERACTIVE	0xA6
A661_LOOP_FORWARD	0x00
A661_LOOP_FORWARD_AND_BLANK	0x02
A661_LOOP_FORWARD_AND_BACKWARD_AND_BLANK	0x03
A661_LOOP_FORWARD_AND_RESET	0x01
A661_MDF_ABSOLUTE	0x65
A661_MDF_BRG_DIST_ACHDG	0x60
A661_MDF_DIST_DIST	0x67
A661_MDF_DIST_DIST_FIX	0x6A
A661_MDF_LAT_LONG	0x61
A661_MDF_LEGACY	0x62
A661_MDF_RELATIVE	0x66
A661_MDF_X_DIST	0x68
A661_MDF_Y_ALT	0x69
A661_MINIMUM_SIZE	0x02
A661_NO_SHUFFLE	0x84
A661_NO_SIZING	0x74
A661_NORMAL	0x1D
A661_NOT_CLAMPED	0x00
A661_NOT_USED	0x00
A661_OPEN_CENTERED	0x17
A661_OPEN_DOWN	0x18
A661_OPEN_UP	0x16
A661_OPERATOR_ADD_ASSIGN_ARG	0x0B
A661_OPERATOR_ADD_ASSIGN_SRC	0x08
A661_OPERATOR_ASSIGNMENT	0x00
A661_OPERATOR_CONSTANT	0x02
A661_OPERATOR_INVERSE	0x01
A661_OPERATOR_LOGICAL_AND	0x06
A661_OPERATOR_LOGICAL_OR	0x07
A661_OPERATOR_PRODUCT	0x05
A661_OPERATOR_PROD_ASSIGN_ARG	0x0D
A661_OPERATOR_PROD_ASSIGN_SRC	0x0A
A661_OPERATOR_SUB_ASSIGN_ARG	0x0C
A661_OPERATOR_SUB_ASSIGN_SRC	0x09
A661_OPERATOR_SUM	0x04
A661_OUT_OF_BOUNDS	0x00
A661_PARALLEL	0x00
A661_PARKING_LINE_START	0xBA
A661_PARKING_LINE_END	0xBB
A661_PIX_FMT_COLOR_INDEXED_8	0xC2
A661_PIX_FMT_LUMINANCE_ALPHA_8	0xC1
A661_PIX_FMT_RGBA_8	0xC0
A661_PLAY	0x01
A661_POPUP_AND_HIGHLIGHT	0x06

ARINC 661 Integer Constant Values	
A661_POPUP_ONLY	0x02
A661_PRESERVE_CHARACTER	**0x01**
A661_PRESERVE_WORD	**0x00**
A661_REPORT_ALL	0x61
A661_REPORT_ALL_WITH_CANDIDATE	**0x64**
A661_REPORT_ON_COMPLETION	0x62
A661_REPORT_ON_TRANSITION	0x60
A661_REPORT_ON_TRANSITION_WITH_CANDIDATE	**0x65**
A661_REPORT_ON_CDS	**0x63**
A661_RIGHT	0x14
A661_RIGHT_TO_LEFT	0x42
A661_RUN	0x01
A661_RUN_ONCE	0x02
A661_SAME_LEVEL	0x02
A661_SELECTED	0x01
A661_SEQUENTIAL	0x01
A661_SEQUENTIAL_FROM	**0x02**
A661_SHUFFLE_DOWN	0x81
A661_SHUFFLE_LEFT	0x82
A661_SHUFFLE_RIGHT	0x83
A661_SHUFFLE_UP	0x80
A661_SIZE_BOTTOM_UP	0x71
A661_SIZE_LEFT_TO_RIGHT	0x72
A661_SIZE_RIGHT_TO_LEFT	0x73
A661_SIZE_TOP_DOWN	0x70
A661_SOURCE_WIDGET_PARAM	0x00
A661_SYMBOL_CIRCLE	0x27
A661_SYMBOL_CIRCLE_INTERACTIVE	0xA7
A661_SYMBOL_GENERIC	0x28
A661_SYMBOL_GENERIC_INTERACTIVE	0xA8
A661_SYMBOL_OVAL	0x2C
A661_SYMBOL_OVAL_INTERACTIVE	0xAC
A661_SYMBOL_PARKABLE	0xB8
A661_SYMBOL_PARKABLE_INTERACTIVE	0xB9
A661_SYMBOL_ROTATED_PARKABLE	**0xBE**
A661_SYMBOL_ROTATED_PARKABLE_INTERACTIVE	**0xBF**
A661_SYMBOL_TARGET_PARKABLE	**0x3D**
A661_SYMBOL_TARGET_PARKABLE_INTERACTIVE	**0xBD**
A661_SYMBOL_RECTANGLE	0x37
A661_SYMBOL_RECTANGLE_INTERACTIVE	0xB7
A661_SYMBOL_ROTATED	0x29
A661_SYMBOL_ROTATED_INTERACTIVE	0xA9
A661_SYMBOL_RUNWAY	0x2A
A661_SYMBOL_RUNWAY_INTERACTIVE	0xAA
A661_SYMBOL_TARGET	0x2D
A661_SYMBOL_TARGET_INTERACTIVE	0xAD
A661_TO	**0x06**
A661_TO_BACK	**0x01**
A661_TO_FRONT	**0x02**
A661_TOP	0x11
A661_TOP_CENTER	0x33

ARINC 661 Integer Constant Values	
A661_TOP_LEFT	0x34
A661_TOP_RIGHT	0x35
A661_TOP_TO_BOTTOM	0x43
A661_TOUCH_ABORT	0x04
A661_TOUCH_DOWN	0x01
A661_TOUCH_MOVE	0x02
A661_TOUCH_MOVE_OUTSIDE	0x05
A661_TOUCH_UP	0x03
A661_TRIANGLE_END	0x32
A661_TRIANGLE_END_DOUBLE	0x33
A661_TRIANGLE_END_DOUBLE_INTERACTIVE	0xB3
A661_TRIANGLE_END_INTERACTIVE	0xB2
A661_TRIANGLE_FAN_START	0x34
A661_TRIANGLE_FAN_START_INTERACTIVE	0xB4
A661_TRIANGLE_SEGMENT	0x2F
A661_TRIANGLE_SEGMENT_DOUBLE	0x31
A661_TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE	0xB1
A661_TRIANGLE_SEGMENT_INTERACTIVE	0xAF
A661_TRIANGLE_STRIP_START	0x2E
A661_TRIANGLE_STRIP_START_INTERACTIVE	0xAE
A661_UNSELECTED	0x00
A661_UP	0x19
A661_VALIDATION	0x01
A661_VALIDATION_AND_WHEEL	0x02
A661_WHEEL	0x03
A661_WRAP_BOTH	0x01
A661_WRAP_FIRST_TO_LAST	0x02
A661_WRAP_LAST_TO_FIRST	0x03
A661_WRAP_NONE	0x00
A661_REP_DEFAULT	0x00
A661_REP_STANDARD	0x01
A661_REP_FOCUS	0x02
A661_REP_HIGHLIGHT	0x03
A661_VERTEX_RENDER_START	0x39
A661_VERTEX_RENDER_SEGMENT	0x3A
A661_VERTEX_RENDER_END	0x3B
A661_VERTEX_RENDER_LINES	0x01
A661_VERTEX_RENDER_LINE_LOOP	0x02
A661_VERTEX_RENDER_POLYLINE	0x03
A661_VERTEX_RENDER_TRIANGLE	0x04
A661_VERTEX_RENDER_TRIANGLE_FAN	0x05
A661_VERTEX_RENDER_TRIANGLE_STRIP	0x06
Reserved for future use	*0xC3 through 0xCF*
Reserved for OEM Customization (for MDF types)	*0xD0 through 0xDF*
Reserved for OEM Customization (for MIL Items types)	*0xE0 through 0xFF*
On 16 bits	
A661_CURSOR_CLICKED	0xB1D3
A661_CURSOR_DOUBLE_CLICKED	0xB1D4

ARINC 661 Integer Constant Values	
A661_CURSOR_PRESSED	0xB1D1
A661_CURSOR_RELEASED	0xB1D2
A661_CURSOR_RIGHT_CLICKED	0xB1D5
A661_RANGE	**0xB1D1**
A661_ORIENTATION	**0xB1D2**
A661_PROJECTION_REFERENCE	**0xB1D3**
A661_STYLE_SET_DEFAULT	0x0000
On 32 bits	
A661_ARC	**0x00000001**
A661_HORZ_LINE	**0x00000002**
A661_VERT_LINE	**0x00000003**

Note: Unlike the other tables, constant values in Table 4.6-11 do not have to be unique.

Table 4.6-12 – Widget Extensions

ARINC 661 Extensions (16 bits)	
Specification range	*0x8000 – 0x87FF*
A661_BOUNDARY_CHECK_EXTENSION	0x8010
A661_CHILD_INDEX_EXTENSION	0x8001
A661_CURSOR_EVENTS_EXTENSION	0x8002
A661_CURSOR_SHAPE_EXTENSION	0x8008
A661_DIRECTIONAL_TABBING_EXTENSION	0x8003
A661_EXCLUDED_REGIONS_EXTENSION	0x800E
A661_FOCUS_STOP_EXTENSION	0x8007
A661_INITIAL_FOCUS_EXTENSION	0x8006
A661_LEGEND_ALIGN_EXTENSION	0x8004
A661_MULTI_SELECTION_EXTENSION	0x800F
A661_PICTURE_EXTENSION	0x8009
A661_REORDER_EXTENSION	**0x8011**
A661_STATIC_PARAM_BUFFER_EXTENSION	0x800D
A661_STYLE_SET_EXTENSION	0x800C
A661_SYMBOL_EXTENSION	0x800A
A661_TICS_ARRAY_EXTENSION	0x800B
A661_VISIBLE_CHILD_EXTENSION	0x8005
A661_WORD_WRAP_EXTENSION	**0x8012**
A661_CURSOR_ENTRY_EVENTS_EXTENSION	**0x8013**
A661_MAPHORZ_COORD_SYSTEM_EVENT_EXTENSION	**0x8014**
A661_MAPVERT_COORD_SYSTEM_EVENT_EXTENSION	**0x8015**
Reserved for future use	*0x8800 through 0x8FFF*

Table 4.6-13 – Character set encodings (16 bits)

ARINC 661 Commands	
Specification range	*0x0000 – 0xB7FF*
A661_ASCII_EXTENDED	0x0001
A661_UTF8	0x0002
A661_GBK	0x0003
Reserved for OEM Customization	*0xB800 to 0xBFFF*

Table 4.6-14 – Animation Laws (16 bits)

ARINC 661 Animation Laws	
Specification range	*0x0000 – 0x7FFF*
A661_LINEAR	0x0000
A661_EASING_IN	0x0001
A661_EASING_OUT	0x0002
A661_EASING_INOUT	0x0003
Reserved for OEM Customization	*0x8000 – 0xFFFF*

Table 4.6-15 – Layouts identifiers (8 bits)

ARINC 661 Layouts Identifiers	
Specification range	***0x00 – 0x7F***
A661_VERTICAL	**0x00**
A661_HORIZONTAL	**0x01**
A661_CIRCULAR	**0x02**
A661_GRID	**0x03**
A661_STAR	**0x04**
Reserved for OEM Customization	***0x80 – 0xFF***

5.0 SYMBOL GRAPHICAL DEFINITION

5.1 Overview

Several widgets (e.g., Symbol widget) and map items (SYMBOL_GENERIC, SYMBOL_ROTATED, and SYMBOL_TARGET) refer to symbols by an ID number ("SymbolReference" or "SymbolType"). A CDS supports a set of predefined symbols. The existing predefined CDS symbology can also be supplemented by symbols defined within DFs. The DF Symbol definitions are found near the start of the DF, before any of the UA Layer definitions. Any UA Layers of a DF can reference the symbols defined in that DF.

The following scheme should be used to find the appropriate symbol for a given symbol ID number:

- If a Symbol with matching ID is found in the DF, use it.
- Otherwise, use the predefined symbol with that ID.

In the case of the SYMBOL_GENERIC, SYMBOL_ROTATED, and SYMBOL_TARGET map items, the legend anchor command of a symbol defines the first legend position.

In the case of the SYMBOL_TARGET map item, there is no command to specify the Length part and the Orientation part of the symbol. Therefore, the way the CDS handles these parameters is implementation dependent. This section describes the Symbol Definition Commands that are used within the Symbol Definition Structure to specify the graphical appearance of the symbol.

- See Section 4.5.3 for the format of the Definition Time structure in general, and the Symbol Definitions Structure in particular.

5.2 Symbol Definition Commands

A single Symbol Definition contains a sequence of Symbol Definition Commands. A Symbol Definition can have up to four different representations, each consisting of graphic primitive and/or attribute commands:

- The standard representation begins the symbol definition.
- An optional focus representation begins with a FOCUS command, and follows the standard representation.
- An optional highlight representation is last, beginning with a HIGHLIGHT command.
- An optional sensitive area representation to define the interactive zone of the symbol.

Commands given in one representation (standard, focus, highlight, or sensitive area) of a symbol do not affect commands given in another representation of that symbol (or any other symbol).

Symbol_definition ::=

symbol_representation

[FOCUS_Command *symbol_representation*]

[HIGHLIGHT_Command *symbol_representation*]

[RECT_AREA_Command | CIRC_AREA_Command]

symbol_representation ::=

{ symbol_attribute_command | symbol_graphic_primitive_command } +

The symbol attribute commands and symbol graphic primitive commands are described in Sections 5.2.2 and 5.2.3 respectively.

5.2.1 Top Level Commands

Table 5.2.1 summarizes the symbol Library.

Table 5.2.1 – Symbol Library Summary

Symbol Type	Description
TOP LEVEL COMMANDS	
SymbolDefnFocus	The FOCUS command identifies the beginning of the definition of the focus representation of the symbol
SymbolDefnHighlight	The HIGHLIGHT command identifies the beginning of the definition of the highlight representation of the symbol
SymbolDefnRectangular	The RECTANGULAR command identifies the definition of the rectangular interactive zone representation of the symbol
SymbolDefnCircular	The CIRCULAR command identifies the definition of the circular interactive zone representation of the symbol.
ATTRIBUTES	
SymbolDefnSetColor	The SET_COLOR command sets the current color index, affecting the color of lines, boundary lines, and fills
SymbolDefnSetLineStyle	The SET_LINE_STYLE command sets (non-color) attributes that effect line (and line segment) drawing: for example, pattern and width.
SymbolDefnSetFont	The SET_FONT command sets the current font to the given font index.
SymbolDefnSetHalo	The SET_HALO command sets the current halo setting. The halo values apply to Symbol graphic primitive commands within a symbol definition the same way as Halo parameters apply to Gp widgets.
TRANSFORMS	
SymbolDefnSetRotation	The SET_ROTATION command sets the current rotation.
SymbolDefnSetTranslation	The SET_TRANSLATION command sets the current translation.
SymbolDefnSetScale	The SET_SCALE command sets the current Scale.
SymbolDefnStartGroup	The START_GROUP command starts a new group of commands.
SymbolDefnEndGroup	The END_GROUP command ends the current group of commands.
PRIMITIVES	
SymbolDefnLegendAnchor	The LEGEND_ANCHOR command explicitly sets the anchor position for the legend attached to the symbol representation.
SymbolDefnArcEllipse	The ARC_ELLIPSE command draws the same shape as the GpArcEllipse widget.
SymbolDefnArcCircle	The ARC_CIRCLE command draws the same shape as the GpArcCircle widget.
SymbolDefnCrown	The CROWN command draws the same shape as the GpCrown widget.

Symbol Type	Description
SymbolDefnLine	The LINE command draws the same shape as the GpLine widget.
SymbolDefnLinePolar	The LINE_POLAR command draws the same shape as the GpLinePolar widget.
SymbolDefnPolyline	The POLYLINE command draws the same shape as the GpPolyline widget.
SymbolDefnRectangle	The RECTANGLE command draws the same shape as the GpRectangle widget.
SymbolDefnTriangle	The TRIANGLE command draws the same shape as the GpTriangle widget.
SymbolDefnTriangleFan	The TRIANGLE_FAN command draws a filled shape like the GpTriangleFan widget.
SymbolDefnTriangleStrip	The TRIANGLE_STRIP command draws a filled shape like the GpTriangleStrip widget.
SymbolDefnText	The TEXT command defines a fixed text string that is part of symbol and is drawn using the current font. The TEXT command does not correspond to any Gp widget.
SymbolDefnSymbolRef	The SYMBOL_REF command allows to draw another Symbol as a command inside a Symbol.
BOUNDING	
SymbolDefnSize (DEPRECATED)	The SIZE command defines the size of a symbol representation. This command was superseded by BOUNDING_RECT and BOUNDING_CIRCLE commands in Supplement 5.
SymbolDefnBoundingRect	The BOUNDING_RECT command defines the bounding box of a symbol representation.
SymbolDefnBoundingCircle	The BOUNDING_CIRCLE command defines the bounding circle of a symbol representation.

5.2.1.1 Focus

The FOCUS command identifies the beginning of the definition of the focus representation of the symbol.

Table 5.2.1.1 – FOCUS Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_FOCUS
UnusedPad	N/A	16	

5.2.1.2 Highlight

The HIGHLIGHT command identifies the beginning of the definition of the highlight representation of the symbol.

Table 5.2.1.2 – HIGHLIGHT Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_HIGHLIGHT
UnusedPad	N/A	16	

5.2.1.3 Sensitive Area

The Sensitive Area command identifies the definition of the interactive zone representation of the symbol. This zone is a rectangular one or a circular one. If this zone is not defined, the behavior is implementation dependent (the symbol may be declared as not interactive, or a default interactive zone may be defined by the CDS).

For a rectangular zone, the parameter “RotationAllowed” is set to A661_TRUE if the Sensitive Area is rotated with the symbol in case of A661_SYMBOL_ROTATED. If this parameter is set to A661_FALSE, the Sensitive area rectangle is always parallel to X and Y axis.

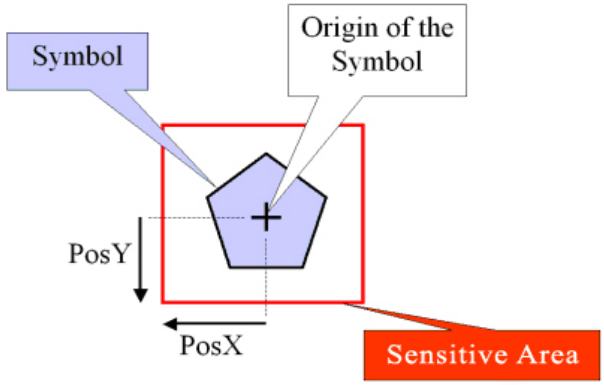


Figure 5.2.1.3 – Sensitive Area

Table 5.2.1.3-1 – Sensitive Area-Rectangular Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_RECTANGULAR
RotationAllowed	N/A	8	A661_TRUE, A661_FALSE
UnusedPad	N/A	8	0
PosX	long	32	The X position: it is the bottom left corner of the rectangle, relative to the symbol origin
PosY	long	32	The Y position: it is the bottom left corner of the rectangle, relative to the symbol origin
SizeX	ulong	32	The width of the Sensitive area.
SizeY	ulong	32	The height of the Sensitive area.

Table 5.2.1.3-2 – Sensitive Area-Circular Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_CIRCULAR
UnusedPad	N/A	16	0
PosX	long	32	The X position of the center of the circle
PosY	long	32	The Y position of the center of the circle
Radius	ulong	32	The radius of the circle.

5.2.2 Symbol Attribute Setting Commands

Symbol Attribute Setting Commands set graphical attributes (Color, Linestyle, Font, Halo) that affect Graphic Primitive Commands.

The StyleSet and/or Color and/or Halo properties of the widget or Map Item that references the symbol determine the initial attribute settings of each of the representations of the symbol. Attribute setting commands (if any) made within a representation of the symbol definition override those initial defaults. The attribute setting commands made within a symbol definition only apply within the context of that symbol: these settings cannot affect subsequent widgets (including other symbols) or map items.

5.2.2.1 Set Color

The SET_COLOR command sets the current color index, affecting the color of lines, boundary lines, and fills. It is an index into the same color table referenced by color index properties of Gp widgets.

Table 5.2.2.1 – SET_COLOR Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_COLOR
ColorIndex	uchar	8	Index into a CDS-defined color table. See Section 3.1.3.3.1.
UnusedPad	N/A	8	0

5.2.2.2 Set Line Style

The SET_LINE_STYLE command sets (non-color) attributes that effect line (and line segment) drawing: for example, pattern and width. The style is an index into a line style table stored on the CDS. Linestyle modulation is reset with every SET_LINE_STYLE command.

Table 5.2.2.2 – SET_LINE_STYLE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_LINE_STYLE
Style	ushort	16	Index into a CDS-defined line style table.

5.2.2.3 Set Font

The SET_FONT command sets the current font to the given font index.

Table 5.2.2.3 – SET_FONT Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_FONT
Font	uchar	8	Index into a CDS-defined font table. See Section 3.1.3.3.1.
UnusedPad	N/A	8	0

5.2.2.4 Set Halo

The SET_HALO command sets the current halo setting. The halo values apply to Symbol graphic primitive commands within a symbol definition the same way as Halo parameters apply to Gp widgets. See Section 3.1.3.3.

Table 5.2.2.4 – SET_HALO Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_HALO
Halo	uchar	8	A661_TRUE A661_FALSE A661_SAME_LEVEL
UnusedPad	N/A	8	0

5.2.3 Coordinate System Commands

The coordinate system commands allow the application of transformations (translations and rotations) to the next set of graphical primitive commands. The StartGroup and EndGroup commands allow the definition of the scope of the transformations.

Note that:

- The limitations of the number of StartGroup commands that can be performed is implementation dependent.
- The result of performing an EndGroup without any previous StartGroup, or to perform more EndGroup than StartGroup which were previously defined, is implementation dependent.

Table 5.2.3 – StartGroup and EndGroup Example

1	**A661_SYMBOL_DEFN_RECTANGLE**	**Not transformed**
2	A661_START_GROUP	Start of first group
3	A661_SYMBOL_DEFN_SET_TRANSLATION	All commands until the end of first group will be translated
4	A661_SYMBOL_DEFN_TEXT	Apply translation 3
5	A661_START_GROUP	Start of second group
6	A661_SYMBOL_DEFN_SET_ROTATION	All commands until the end of second group will be rotated
7	A661_SYMBOL_DEFN_ARC_CIRCLE	Apply translation 3 and rotation 6
8	A661_END_GROUP	End of second group: end of rotation 6
9	A661_SYMBOL_DEFN_TEXT	Apply translation 3
10	A661_END_GROUP	End of first group: end of translation 3
11	A661_SYMBOL_DEFN_TEXT	Not transformed

5.2.3.1 Rotation

The SET_ROTATION command sets the current rotation. It applies to all the next graphical commands until the end of the Symbol representation or the end of the current group.

Table 5.2.3.1 – SET_ROTATION Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_ROTATION
UnusedPad	N/A	16	0
RotationAngle	fr(180)	32	Rotation angle value to be applied to the next primitive commands
CenterX	long	32	X position of the center of the rotation
CenterY	long	32	Y position of the center of the rotation

5.2.3.2 Translation

The SET_TRANSLATION command sets the current translation. It applies to all the next graphical commands until the end of the Symbol representation or the end of the current group.

Table 5.2.3.2 – SET_TRANSLATION Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_TRANSLATIO N
UnusedPad	N/A	16	0
TranslationX	long	32	X translation
TranslationY	long	32	Y translation

5.2.3.3 Scale

The SET_SCALE command sets the current Scale. It applies to all the next graphical commands until the end of the Symbol representation or the end of the current group.

Note: The result of scaling text is implementation dependent.

Example:

Index	Symbol Command	Effect
1	A661_SYMBOL_DEFN_SET_TRANSLATION	All commands until the end of the Symbol will be translated
2	A661_SYMBOL_DEFN_RECTANGLE	Apply translation 1
3	A661_START_GROUP	Start of first group
4	A661_SYMBOL_DEFN_SET_SCALE	All commands until the end of the Group will be scaled
5	A661_SYMBOL_DEFN_TEXT	Apply scale 4, then translation 1
6	A661_SYMBOL_DEFN_RECTANGLE	Apply scale 4, then translation 1
7	A661_END_GROUP	End of first group: end of scale 4
8	A661_SYMBOL_DEFN_TEXT	Apply translation 1

Table 5.2.3.3 – SET_SCALE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SET_SCALE
UnusedPad	N/A	16	0
ScaleX	float	32	Scale of the Symbol commands on the X axis
ScaleY	float	32	Scale of the Symbol commands on the Y axis

5.2.3.4 Start Group

The START_GROUP command starts a new group of commands.

Table 5.2.3.4 – START_GROUP Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_START_GROUP
UnusedPad	N/A	16	0

5.2.3.5 End Group

The END_GROUP command ends the current group of commands.

Table 5.2.3.5 – END_GROUP Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_END_GROUP
UnusedPad	N/A	16	0

5.2.4 Graphic Primitive Commands

This section describes symbol commands to define the graphic primitives that make up a symbol.

All coordinate, height, width, and radius arguments are in units of hundredths of mm. All angles are in degrees and follow the rules specified in Section 2.3.4.2, Angles. The (0,0) position of the symbol represents the part of the symbol that will be placed at the widget or map item's position. Symbols are sensitive to rotation and/or translations that are applied by ancestor widgets.

5.2.4.1 Legend Anchor

The LEGEND_ANCHOR command explicitly sets the anchor position for the legend attached to the symbol representation. The anchor position can apply to one or more types of legends (for example, the same position can be applied to both a LEGEND_POP_UP and a LEGEND_HIGHLIGHT). If there is more than one Legend Anchor command associated with the same type of legend, the behavior is implementation dependent. If there is no LEGEND_ANCHOR command within a symbol representation, the legend anchor position is (0,0) for all legend types. The LEGEND_ANCHOR command does not correspond to any Gp widget.

Below are examples illustrating how legend anchors and legend map item types can be used. Note that the order in which different legend types appear in the map buffer do not need to correspond to order of legend anchors in the symbol definition.

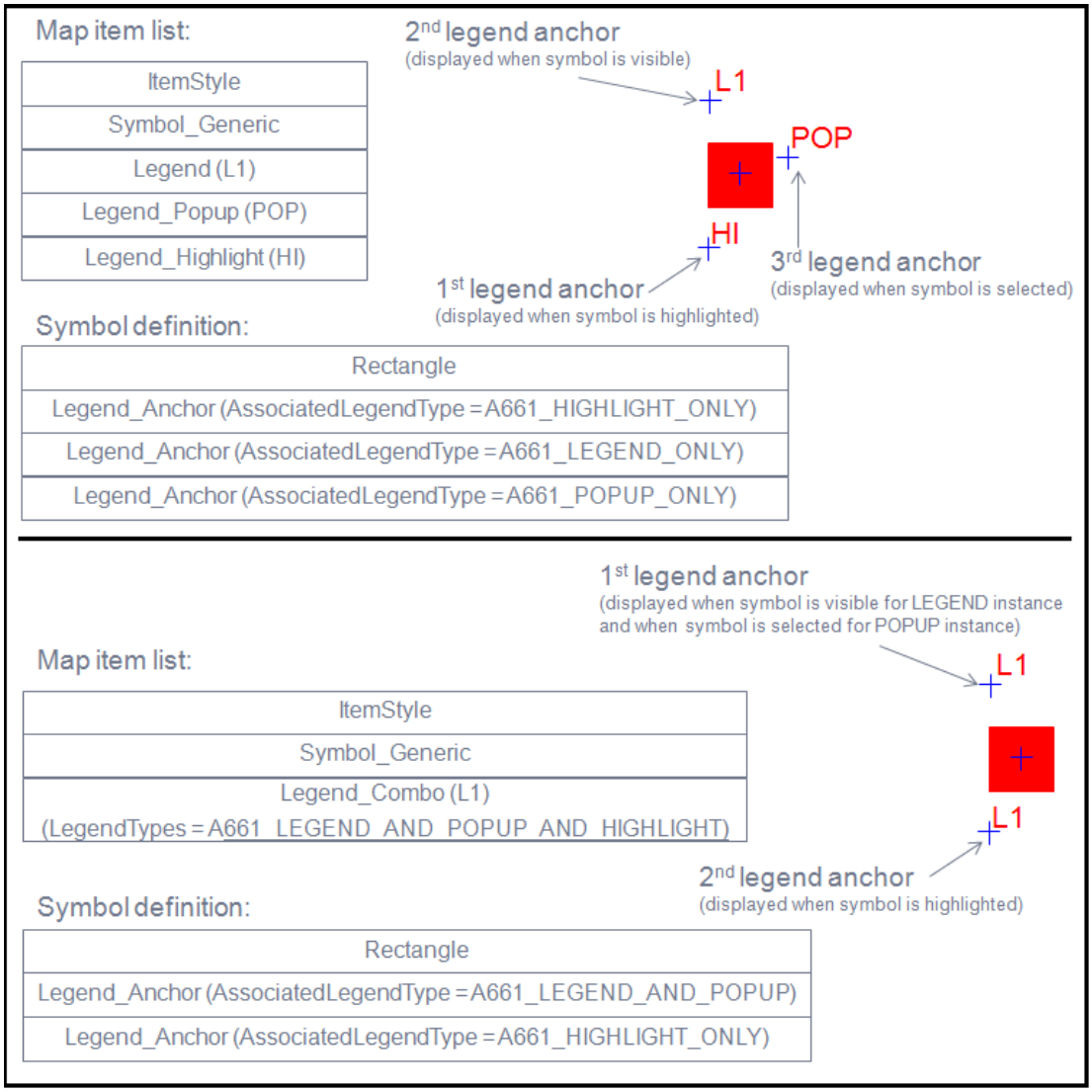


Figure 5.2.4.1 – Legend_Anchor examples

Table 5.2.4.1 – LEGEND_ANCHOR Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_LEGEND_ANCHOR
UnusedPad	N/A	8	0
AssociatedLegendType	uchar	8	Defines the legend types for which this anchor position is used. A661_LEGEND_ONLY A661_POPUP_ONLY A661_HIGHLIGHT_ONLY A661_LEGEND_AND_POPUP A661_LEGEND_AND_HIGHLIGHT A661_POPUP_AND_HIGHLIGHT A661_LEGEND_AND_POPUP_AND_HIGHLIGHT Value of 0 is reserved.
PosX	long	32	The X position.
PosY	long	32	The Y position.

5.2.4.2 Arc Ellipse

The ARC_ELLIPSE command defines an arc shape that may be a portion of an ellipse or an arc. It is defined by a bounding box where a rectangle is specified and the ellipse is drawn touching the rectangle. When the bounding box is a square, the arc will be a circle. The major and minor axes of the ellipse are implicitly along the cardinal directions of the bounding box.

The ARC_ELLIPSE command draws the same shape as the GpArcEllipse widget. As in the GpArcEllipse widget, the ARC_ELLIPSE becomes a complete ellipse or circle when the StartAngle and EndAngle represent the minimum and maximum possible values of a 32-bit fr(180), corresponding to -180 degrees and slightly less than +180 degrees, respectively.

The ARC_ELLIPSE is drawn using the current color (in both the filled and unfilled cases). If it is unfilled, the outline is drawn using the current linestyle.

Table 5.2.4.2 – ARC_ELLIPSE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_ARC_ELLIPSE
Filled	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	0
PosX	long	32	The X start position of the bounding box (lower left corner).
PosY	long	32	The Y start position of the bounding box (lower left corner).
SizeX	ulong	32	The maximum width the ARC_ELLIPSE can have (if its StartAngle and EndAngle defines a full ellipse).
SizeY	ulong	32	The maximum height the ARC_ELLIPSE can have (if its StartAngle and EndAngle defines a full ellipse).
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the ARC_ELLIPSE.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the ARC_ELLIPSE.

5.2.4.3 Arc Circle

The ARC_CIRCLE command can define either an arc or a circle. It is defined by a center position, a start angle, and an end angle. The arc sweeps counterclockwise from the start angle to the end angle.

The ARC_CIRCLE command draws the same shape as the GpArcCircle widget.

The following figure illustrates the two ARC_CIRCLE command cases, depending on the Filled setting. The small circles are for reference purposes and indicate the circle centers (they are not actually drawn). The ARC_CIRCLE is drawn using the current color (in both the filled and unfilled cases). If it is unfilled, the outline is drawn using the current linestyle.

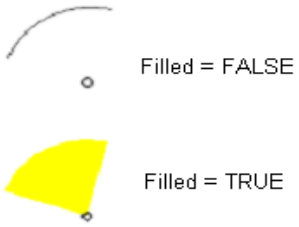


Figure 5.2.4.3 – Arc Circle

Table 5.2.4.3 – ARC_CIRCLE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_ARC_CIRCLE
Filled	uchar	8	A661_TRUE A661_FALSE
UnusedPad	N/A	8	0
PosX	long	32	The center X position of the arc or circle.
PosY	long	32	The center Y position of the arc or circle.
Radius	ulong	32	The radius of the arc or circle.
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the arc or circle.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the arc or circle.

5.2.4.4 Crown

The CROWN command defines a 2D doughnut-shaped region, defined by a center, two radii, a StartAngle, and an EndAngle. The crown sweeps the region going counterclockwise from the start angle to the end angle. The crown becomes closed to form a complete “doughnut” shape when the StartAngle and EndAngle represent the minimum and maximum possible values of a 32-bit fr(180), corresponding to -180 degrees and slightly less than +180 degrees, respectively.

A crown can be filled or unfilled (outlined). The following diagram shows an outlined crown. The crown is drawn using the current color (in both the filled and unfilled cases). If it is outlined, the outline is drawn using the current linestyle.

The CROWN command draws the same shape as the GpCrown widget.

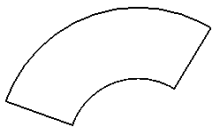


Figure 5.2.4.4 – Crown

Table 5.2.4.4 – CROWN Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_CROWN
Filled	uchar	8	A661_TRUE A661_FALSE.
UnusedPad	N/A	8	0
PosX	long	32	The center X position of the crown.
PosY	long	32	The center Y position of the crown.
InnerRadius	ulong	32	The inner radius.
OuterRadius	ulong	32	The outer radius.
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the crown.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the crown.

5.2.4.5 Line

The LINE command draws a line between the two positions specified. The LINE is drawn using the current color and linestyle. The LINE command draws the same shape as the GpLine widget.

Table 5.2.4.5 – LINE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_LINE
UnusedPad	N/A	16	0
PosXStart	long	32	The start X position of the line.
PosYStart	long	32	The start Y position of the line.
PosXEnd	long	32	The end X position of the line.
PosYEnd	long	32	The end Y position of the line.

5.2.4.6 Line Polar

The LINE_POLAR command defines a line in polar coordinates with an X,Y coordinate start position, a line length, and a draw angle. The LINE_POLAR is drawn using the current color and linestyle. The LINE_POLAR command draws the same shape as the GpLinePolar widget.

Table 5.2.4.6 – LINE_POLAR Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_LINE_POLAR
UnusedPad	N/A	16	0
PosXStart	long	32	The start X position of the line.
PosYStart	long	32	The start Y position of the line.
RotationAngle	fr(180)	32	The angle at which the line is drawn.
LineLength	long	32	The length of the line.

5.2.4.7 Polyline

The POLYLINE command defines a sequence of connected lines. It is drawn using the current color and linestyle.

The following figure shows an example of how an unclosed polyline defined using vertices (V0 ... V6) could be defined (when drawn, the polyline will not show the (V0...V6) labels).

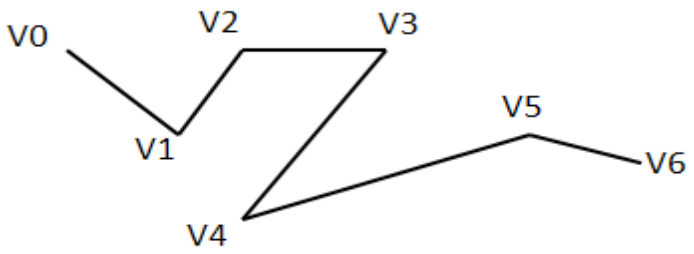


Figure 5.2.3.7 – Polyline

Table 5.2.4.7 – POLYLINE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_POLYLINE
NumVertices	ushort	16	number of vertices (minimum of 3 vertices required)
Closed	uchar	8	A661_TRUE: an extra line is drawn from the last vertex to the first vertex. A661_FALSE: no extra line is drawn from the last vertex to the first vertex.
UnusedPad	N/A	24	0
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.4.8 Rectangle

The RECTANGLE command defines a rectangle, according to a lower left corner position, a height, and a width. The RECTANGLE is drawn using the current color (in both the filled and unfilled cases). If unfilled, it uses the current linestyle. The RECTANGLE command draws the same shape as the GpRectangle widget.

Table 5.2.4.8 – RECTANGLE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_RECTANGLE
Filled	uchar	8	**A661_TRUE** **A661_FALSE**
UnusedPad	N/A	8	0
PosX	long	32	The X start position of the rectangle (lower left corner).
PosY	long	32	The Y start position of the rectangle (lower left corner).
SizeX	ulong	32	The width of the rectangle.
SizeY	ulong	32	The height of the rectangle.

5.2.4.9 Triangle

The TRIANGLE command defines a triangle, according to three defined vertices. The TRIANGLE is drawn using the current color (in both the filled and unfilled cases). If unfilled, it uses the current linestyle. The TRIANGLE command draws the same shape as the GpTriangle widget.

Table 5.2.4.9 – TRIANGLE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_TRIANGLE
Filled	uchar	8	**A661_TRUE** **A661_FALSE**
UnusedPad	N/A	8	0
PosX	long	32	The X start position of the triangle.
PosY	long	32	The Y start position of the triangle.
PosX2	long	32	The X position of the second vertex of the triangle.
PosY2	long	32	The Y position of the second vertex of the triangle.
PosX3	long	32	The X position of the third vertex of the triangle.
PosY3	long	32	The Y position of the third vertex of the triangle.

5.2.4.10 Triangle_Fan

The TRIANGLE_FAN command defines a filled shape composed out of a fan of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the first and last vertices of the previous triangular section. At least three vertices must be specified. Any convex polygon can be represented as a triangle fan, by just specifying its vertices in the natural order. A triangle fan is not necessarily a convex polygon, though. It is drawn using the current color.

The following figure shows how the Vertices (V0 ..V6) of a triangle fan define a filled shape formed out of triangles. When drawn, a triangle fan does not draw the lines or vertex labels shown in this figure. In this case it defines a concave polygon:

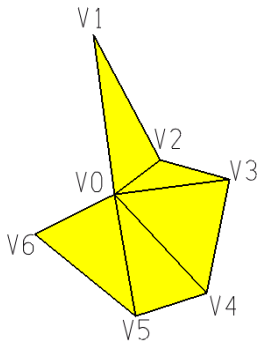


Figure 5.2.4.10-1 – Triangle Fan – Concave Polygon

The following figure illustrates a convex polygon case:

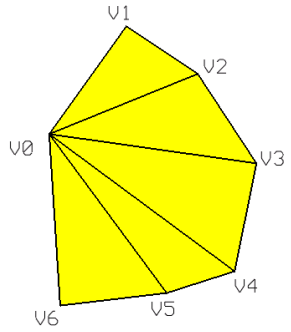


Figure 5.2.4.10-2 – Triangle Fan – Convex Polygon

Table 5.2.4.10 – TRIANGLE_FAN Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_TRIANGLE_FAN
NumVertices	ushort	16	Number of vertices, must be >= 3.
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.4.11 Triangle_Strip

The TRIANGLE_STRIP command defines a filled shape composed out of a linked strip of triangles. The first three vertices define the first triangular section. Each subsequent vertex defines a new triangular section, sharing the last two vertices of the previous triangular section. It is drawn using the current color.

The following figure shows how the Vertices (V0 ..V5) of a triangle strip define a filled shape formed out of triangles. When drawn, a triangle fan does not draw the lines or vertex labels shown in this figure.

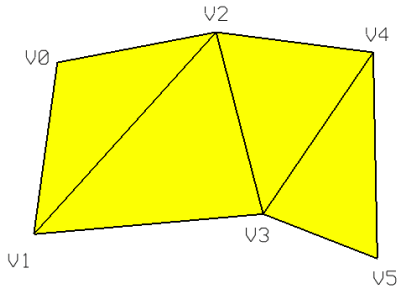


Figure 5.2.4.11 – Triangle Strip

Table 5.2.4.11 – TRIANGLE_STRIP Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_TRIANGLE_STRIP
NumVertices	ushort	16	Number of vertices, must be >= 3.
Vertices	array of (long, long)	64 * NumVertices	An array of (X,Y) pairs.

5.2.4.12 Text

The TEXT command defines a fixed text string that is part of symbol and is drawn using the current font. The first character of the text string is placed according to the PosX, PosY, and Alignment (see Section 3.2.5.6) and subsequent characters are drawn to the right. It is drawn using the current color. The effect of current linestyle on Text for a particular font is implementation dependent. The TEXT command does not correspond to any Gp widget.

Table 5.2.4.12 – TEXT Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_TEXT
Alignment	uchar	8	specifies the alignment of the text with respect to PosX and PosY. A661_TOP_LEFT A661_TOP_CENTER A661_TOP_RIGHT A661_LEFT A661_CENTER A661_RIGHT A661_BOTTOM_LEFT A661_BOTTOM_CENTER A661_BOTTOM_RIGHT
UnusedPad	N/A	8	0
PosX	long	32	The X coordinate of the text anchor.
PosY	long	32	The Y coordinate of the text anchor.
RotationAngle	fr(180)	32	Specifies rotation of text with respect to PosX and PosY.
StringText	string	8 * string length + pad	Null-terminated string, followed by zero, one, two, or three extra NULL for 32 bits alignment.

5.2.4.13 Size (DEPRECATED)

This command was superseded by BOUNDING_RECT and BOUNDING_CIRCLE commands in Supplement 5.

The SIZE command defines the size of a symbol representation. If multiple SIZE commands are found in a representation, all but the last are ignored. If there is no SIZE command in a symbol representation then a default size of zero is used for that representation.

Size defines a bounding box which does not change if the symbol is rotated. Symbols are not clipped to the SizeX/SizeY region defined by this command.

Table 5.2.4.13 – SIZE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SIZE
UnusedPad	N/A	16	0
SizeX	ulong	32	The width of the symbol representation.
SizeY	ulong	32	The height of the symbol representation.

5.2.4.14 Bounding Rectangle

The BOUNDING_RECT command defines the bounding box of a symbol representation. If multiple BOUNDING_RECT commands are found in a representation, all but the last are ignored. If there is no BOUNDING_RECT command in a symbol representation then the definition of the bounding box is implementation dependent.

The OffsetX and OffsetY parameters define the lower left corner of the bounding box; its width and height are defined by SizeX and SizeY. The bounding box may be used by widgets such as SymbolPushButton to align a symbol within a rectangular region; therefore, it should enclose the entire symbol graphical definition and may include a margin (empty space around the graphical definition).

Symbols are not clipped to the region defined by the SizeX/SizeY parameters of this command.

Table 5.2.4.14 – BOUNDING_RECT Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_BOUNDING_RECT
UnusedPad	N/A	16	0
OffsetX	long	32	The left edge of the symbol representation, with respect to origin.
OffsetY	long	32	The right edge of the symbol representation, with respect to origin.
SizeX	ulong	32	The width of the symbol representation.
SizeY	ulong	32	The height of the symbol representation.

5.2.4.15 Bounding Circle

The BOUNDING_CIRCLE command defines the bounding circle of a symbol representation. If multiple BOUNDING_CIRCLE commands are found in a representation, all but the last are ignored. If there is no BOUNDING_CIRCLE command in a symbol representation then the definition of the bounding circle is implementation dependent.

The bounding circle, centered on the origin, is defined by the Radius parameter. It may be used by map widgets (which may rotate a symbol) to perform visibility testing; therefore, it should enclose the entire symbol graphical definition.

Table 5.2.4.15 – BOUNDING_CIRCLE Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_BOUNDING_CIRCLE
UnusedPad	N/A	16	0
Radius	ulong	32	The radius of the bounding circle of the symbol representation, centered on the origin.

5.2.4.16 Symbol Reference

The SYMBOL_REF command allows to draw another Symbol as a command inside a Symbol. Note that the current LineStyle and Color of the enclosing Symbol applies to the symbol reference. Also, the Rotation, Translation, and Scale commands will be applied to the Symbol reference.

The Representation parameter allows to define which representation to use for the Symbol reference:

A661_REP_DEFAULT: the same representation as the calling Symbol Command

A661_REP_STANDARD: the standard representation

A661_REP_FOCUS: the FOCUS representation

A661_REP_HIGHLIGHT: the HIGHLIGHT representation

The effect of referencing another symbol is equivalent to replacing the reference by the commands present in the referenced symbol representation.

Note: If you want to protect further commands defined after the SYMBOL_REF command from rotation, translation, or scale changes, you should enclose explicitly the symbol referenced by the SYMBOL_REF by a Start Group and End Group.

The maximum depth of nested A661_SYMBOL_DEFN_SYMBOL_REF commands is implementation dependent.

The following example shows an equivalent symbol with and without the set symbol reference command:

Index	Symbol Command	Effect
1	A661_CMD_CREATE_SYMBOL[1]	New Symbol with Symbol ID 1
2	A661_SYMBOL_DEFN_SET_TRANSLATION	Apply translation 2
3	A661_SET_COLOR	Set Color 3
4	A661_SYMBOL_DEFN_RECTANGLE	Apply translation 2, and Color 3
…		
5	A661_CMD_CREATE_SYMBOL[2]	New Symbol with Symbol ID 2
6	A661_SYMBOL_DEFN_SET_ROTATION	Apply rotation 6
7	A661_SYMBOL_DEFN_TEXT	Apply rotation 6
8	A661_SYMBOL_DEFN_SYMBOL_REF[1]	Insert symbol reference 1 here
9	A661_SYMBOL_DEFN_ARC_CIRCLE	Apply translation 2, then rotation 6, and Color 3

The effect of this definition for Symbol ID 2 will be equivalent to:

Index	Symbol Command	Effect
6	A661_SYMBOL_DEFN_SET_ROTATION	Apply rotation 6
7	A661_SYMBOL_DEFN_TEXT	Apply rotation 6
8.1	A661_SYMBOL_DEFN_SET_TRANSLATION	Apply translation 8.1, then rotation 6
8.2	A661_SET_COLOR	Set Color 8.2
8.3	A661_SYMBOL_DEFN_RECTANGLE	Apply translation 8.1, then rotation 6, and Color 8.2
9	A661_SYMBOL_DEFN_ARC_CIRCLE	Apply translation 8.1, then rotation 6, and Color 8.2

Table 5.2.4.16 – SYMBOL_REF Command Structure

Field Name	Type	Size (bits)	Description
SymbolCommandType	ushort	16	A661_SYMBOL_DEFN_SYMBOL_REF
SymbolReference	ushort	16	The reference of the Symbol to draw
Representation	uchar	8	Specifies the representation to use for the reference A661_REP_DEFAULT A661_REP_STANDARD A661_REP_FOCUS A661_REP_HIGHLIGHT
UnusedPad	N/A	24	0
PosX	long	32	The X coordinate of the Symbol reference.
PosY	long	32	The Y coordinate of the Symbol reference.

6.0 XML DEFINITION FILE SPECIFICATION

6.1 Introduction

This section describes how an ARINC 661 definition file is defined in Extensible Markup Language (XML) format. It is assumed that the reader is familiar with XML and ARINC 661 widgets.

The XML DF format is intended to provide a convenient interchange format between ARINC 661 tools. It also has the advantage that it is human-readable. The XML DF format includes all the information found in the binary DF, but may also contain additional information. For example, the XML DF optionally allows names to be given to widgets (in addition to the widget ID). This is convenient for various ARINC 661 tools (e.g., UA Layer editors). There is also provision for additional implementation dependent information to be added (which may not correspond to information found in the binary DF).

Since the XML DF can contain more information than a binary DF, a round-trip conversion from an XML DF to a binary DF and back again could lose information.

The XML DF grammar description provided defines the core grammar subset. It does not describe the full grammar because it is expected that implementation dependent XML elements will also be added.

The XML DF grammar is not widget-type-specific, so does not need to be changed to support new widgets.

Refer to Appendix C for XML DF examples.

6.1.1 XML Definition File XSD (XML Schema Document)

This schema defines the interface syntax, superseding the DTD equivalent, which is deprecated in Supplement 6. The schema is available as an electronic support file within the zip at URL:

https://www.aviation-ia.com/support-files/661-7-df

The zip also contains associated examples.

Rules for an ARINC 661 XML DF to be compliant with a particular version of ARINC 661:

1. It should have a DTD (as specified by the DOCTYPE element) or refer to a Schema, and be both well-formed and valid (i.e., it has to conform to that DTD or this Schema).
2. The Schema definition should only differ from the ARINC 661 Standard Schema in the following ways:
   - It can introduce new XML elements that use a different namespace.
   - It can introduce those new XML elements as possible children of existing XML elements.
   - It can add new attributes that use a different namespace to existing XML elements.
   - It can have other syntactic differences that do not affect content. This includes (but is not limited to) differences in comments or whitespace.
3. Attribute names should match the widget parameter names appearing elsewhere in this document (see Sections 3.1.3, 3.3, 3.4, 3.5, 3.6, 3.7, etc.).

In other words, the spelling of attribute names and widget parameter names should match.

A tool loading the ARINC 661 XML DF:

- Should reject any file that is not both “valid” and “well-formed” in the standard XML sense. In other words, the file has to have good XML syntax and must conform to its DTD to be accepted.
- Has undefined behavior if the ARINC 661 XML DF is not ARINC 661 compliant, as defined in the rules above.
- Should skip any of the following, possibly giving warnings:

a. Any unrecognized XML element (including children).
b. Any unrecognized XML attribute of a standard ARINC 661 XML element.
c. Any standard A661 XML element that it does not support (including children). Here are examples:
    - Any **property** XML element (i.e., **prop**, **structprop**, or **arrayprop**) whose (property) name attribute value is not supported, as well as any children it might have.
    - Any **symboldefncmd** XML element whose **type** attribute value is not supported, as well as any children it might have.
    - Any **a661_widget** XML element whose **type** attribute value is not supported, as well as any children it might have).
    - Any **a661_extension** XML element whose **type** attribute value is not supported, as well as any children it might have.

6.2 Description

The following XML fragment gives an example of how model XML elements are used to define properties. In it a model XML element specifies the parameter values of a CheckButton widget:

```
<a661_widget name="CKB_CHOICE" type="A661_CHECK_BUTTON">
     <model>
       <prop name="WidgetIdent" value="2"/>
       <prop name="Visible" value="A661_TRUE"/>
       <prop name="Enable" value="A661_TRUE"/>
       <prop name="CheckButtonState" value="A661_UNSELECTED"/>
       <prop name="StyleSet" value="1"/>
       <prop name="PosX" value="100"/>
       <prop name="PosY" value="100"/>
       <prop name="SizeX" value="3000"/>
       <prop name="SizeY" value="900"/>
       <prop name="NextFocusedWidget" value="0"/>
       <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
       <prop name="LabelString" value="SELECT"/>
       <prop name="MaxStringLength" value="7"/>
       <prop name="Alignment" value="A661_LEFT"/>
       <prop name="PicturePosition" value="A661_LEFT"/>
    </model>
 </a661_widget>
```

6.2.1 Charset Encoding

The optional charset-encoding XML element (not to be confused with the top-level <?xml encoding="…" ?>) specifies the character set encoding for User Application DF strings with its “name” attribute. Section 3.2.5.1 describes how the character set encoding specified affects User Application DF strings.

6.2.1.1 ASCII Extended Character Mappings

As described in Section 3.2.5.1:

- Different implementations may define the ASCII Extended character set differently. For example, there can be implementation dependent characters.
- Appendix L defines an XML format which formalizes the definition of the ASCII Extended character mapping used for a particular implementation.

In addition to specifying the character encoding of the DF, the charset-encoding XML element also has an ascii_extended_mapping attribute:

- This optional attribute can be used to specify the name of the ASCII Extended character set being used, which might be formalized according to Appendix L XML.
- This attribute is only applicable when the character set encoding is A661_ASCII_EXTENDED.

The purpose of the ascii_extended_mapping attribute is to ease exchange of XML DFs between CDS implementations where different ASCII Extended character mappings might be used. If an XML DF with character set encoding of ASCII Extended is to be imported to a CDS having a different ASCII Extended character mapping, the two ASCII Extended character mapping files provide enough information to allow conversion of the imported XML DF into a compatible binary DF for that CDS. There are also UA considerations when importing XML Definition Files having different ASCII Extended character mappings.

6.2.2 Symbol Graphical Definitions, Picture Definitions, and Animation Law Definitions

XML elements related to Symbol Graphical Definitions, Picture Definitions, and Animation Law Definitions can be defined locally within the DF. See Section 5 for more information on Symbol Definitions. See Section 7 for more information on Picture Definitions. See Section 9 for more information on Animation Law Definitions.

The XML DF defines pictures a bit differently than the binary DF. Refer to Section 7 for more details on the binary representation. What follows is a quick comparison.

The binary DF has zero or more **A661_Picture_Block_Structure_DT** structures, each of which specifies:

- **PixelFormat**: A661_PIX_FMT_RGBA_8, A661_PIX_FMT_LUMINANCE_ALPHA_8, A661_PIX_FMT_COLOR_INDEXED_8, or implementation dependent value.
- NumberOfPicturesInBlock.
- NumberOfColorTableEntries.
- **ColorTableFormat**: A661_PIX_FMT_RGBA_8, A661_PIX_FMT_LUMINANCE_ALPHA_8 , or implementation dependent value. (Only useful if **NumberOfColorTableEntries** is greater than 0 and the PixelFormat references a color palette, e.g., A661_PIX_FMT_COLOR_INDEXED_8).
- The color table, composed of **NumberOfColorTableEntries** color table entries, whose representation depends on the **ColorTableFormat.**
- The picture definitions, each of which specifies:

- PictureReference.
- NumberOfPixelsWidth.
- NumberOfPixelsHeight.
- The pixel data of the picture according to the **PixelFormat** (and references the color table if applicable).

The XML DF has an optional **picturetable** XML element, which contains:

- Zero or more **picturedefn** XML elements, each of which specifies:
  - A PictureReference property.
  - A **PixelFormat** property.
  - A ColorTableFormat property, only used if **PixelFormat** is color palette-based (e.g., A661_PIX_FMT_COLOR_INDEXED_8).
  - An **ImageFile** property giving the name of a PNG file.

When converting from an XML DF to a binary DF:

- The pixel data is found in the image file. It might need to be transformed before copying to the binary DF, e.g., if it is not represented in the required **PixelFormat**.
- The **NumberOfPixelsWidth** and **NumberOfPixelsHeigh**t information is found in the image file.
- The color tables are computed as necessary based on the required **ColorTableFormat** and the colors used in the image files.
- **A661_Picture_Block_Structure_DT** structures are created where necessary, from **picturedefn** XML elements having the same PixelFormat (and **ColorTableFormat** if applicable). The maximum size of the color tables is limited by the **PixelForma**t. This may limit which pictures may be grouped together. For example, pictures using **PixelFormat** of A661_PIX_FMT_COLOR_INDEXED_8 can only be grouped together if the total number of colors they use does not exceed 256. The exact details of this algorithm are implementation dependent. The main goal is to reduce binary DF size by grouping pictures using the same color palette format together, where possible.
- The prop, structprop, and arrayprop XML elements specify the value of a property, having a simple, structure, or array type, respectively
- The field, structfield and arrayfield XML elements specify the value of a field of a structure, having a simple, structure, or array type, respectively
- The entry, xyentry, structentry, and arrayentry XML elements specify the value of an entry of an array, having a simple, (x,y) coordinate, structure, or array type, respectively. Note that it is valid to specify a simple (x,y) coordinate by either a xyentry or a structentry with “x” and “y” fields.

6.2.3 Layers and Widgets

This section deleted in Supplement 6. Content deleted in Supplement 6 is now available in the electronic support files.

6.2.4 Properties

Here is a summary of the XML elements described in this section:

The model XML element holds the property values of a particular object.

- The prop, structprop, and arrayprop XML elements specify the value of a property, having a simple, structure, or array type, respectively.
- The field, structfield and arrayfield XML elements specify the value of a field of a structure, having a simple, structure, or array type, respectively.
- The entry, xyentry, structentry, and arrayentry XML elements specify the value of an entry of an array, having a simple, (x, y) coordinate, structure, or array type, respectively. Note that it is valid to specify a simple (x, y) coordinate by either an xyentry or a structentry with “x” and “y” fields.

In principle, these elements can be nested deeply, to allow arrays of structures of arrays, etc. In practice, the usage is much more limited, based on the much simpler data types which are used within the ARINC 661 standard.

If an object has a property, but its value is not specified within the model element, an implementation dependent default value of that property should be used instead.

6.2.4.1 General Property Considerations

- Properties that contain a variable number of elements should be modeled using an array. The property that specifies the size of the array should be specified before the array property itself. Here are some examples:
  - The EntryList parameter of the ComboBox widget
  - The PopUpIdentArray, EnableArray and StringArray parameters of the PopUpMenu or PopUpMenuButton widgets
- Properties that specify an enumerated type should use the name of the enumerated values found in Section 4 (i.e., use the same spelling as defined in Section 4). For example, Boolean properties should show values “A661_TRUE” and “A661_FALSE” in the XML file.
- Properties that provide an attribute index (color, styleset, symbol, or linestyle) can be given either as a symbolic name, or as a numeric index. The symbolic names are implementation dependent.
  - In the case of a symbol index, if it refers to a locally defined symbol, the name of the symbol as given in the symboldefn XML item can be used. The name of a CDS-defined implementation dependent symbol could also be used.
- Properties that specify an “fr” (fixed real) value can be expressed in either of the following ways
  - As a “real” value if the number given contains a “.”
  - As an integral representation of the fixed real value if the number given does not contain a “.”
- Handling a “fr” property whose value is outside its encoding range (example: “200.0” for a fr(180) property) is implementation dependent.
- For integral numeric values, a hex representation should also be supported. A hex value starts with zero, followed by the lowercase letter “x”, followed by a sequence of uppercase hex digits (e.g., “0x6F12”).
- The UTF-8 XML encoding is used in all ARINC 661 XML files. UTF-8 is a multi-byte encoding of UNICODE which uses one byte to directly encode UNICODE characters with codes 127 or below (corresponding to ASCII), and two or more bytes for other UNICODE character cases. Characters can

also be specified via XML entity specifications (e.g., “&”) and “\xHH” ARINC 661-specific backslash-escaping (see Section 6.2.4.2) to avoid that “direct encoding”. XML character references (e.g., “&”) are also legal, but would typically not be used in the ASCII Extended case, since that syntax cannot be specified via the ASCII Extended character mapping files described in
Appendix L.

- Refer to Section 3.2.5.1 for the description of the ARINC 661 ASCII Extended character set, which introduces ARINC 661 specific characters, and also leaves many characters implementation dependent.

- The binary representation of the DefaultStyleText property is defined as: TOutline⊗TBackColor⊗TForeColor⊗TFont. Refer to Table 3.2.5.5-1 for the byte representations of TOutline, TBackColor, TForeColor and TFont. A NULL representation is also possible. The DefaultStyleText property can have any of the following representations in XML. In the first two XML representations, all binary bytes are explicitly set by XML element ‘value’ attributes. In the last XML representation, the binary representation contains byte values that are not directly given by XML element ‘value’ attributes.
  - Using a prop XML element, with the “value” attribute giving the byte values using a string representation:
    - Backslash-escaping is used to represent the 12 byte values as characters. See Section 6.2.4.2.
    - The NULL case is represented with an empty string.
  - Using an arrayprop XML element:
    - The value attribute of each entry XML element child specifies a single byte value.
    - The NULL case is represented with an arrayprop XML element having a single entry XML child with value=”0”.
  - Using a structprop XML element, where field XML element children provides the values.
    - The value attribute of the child XML element:
    - With name=“toutline” provides the Tvalue0 value within toutline.
    - With name=“tbackcolor” provides the Tvalue1 within tbackcolor.
    - With name=“tforecolor” provides the Tvalue1 within tforecolor.
    - With name=“tfont” provides the Tvalue2 within tfont.
    - The NULL case is represented by setting all of TOutline, TBackColor, TForeColor and TFont to -1.

6.2.4.2 Backslash-Escaping

An ARINC-661-specific backslash-escaping mechanism is provided. \xHH specifies a single character, where “HH” represents two hex digits. The “x” must be in lowercase, the hex letter digits ‘A’ through ‘F’ must be given in uppercase, and there must be exactly two hex digits.

Usages of \xHH backslash-escaping:

- Strongly recommended to use for characters within the context of ARINC 661 escape sequences, where the “characters” represent byte values interpreted in an ARINC 661-specific way (see Section 3.2.5.5).

- In the ASCII Extended character encoding case, they can be used outside ARINC 661 escape sequences (e.g., to specify non-standard custom characters). See Appendix L.
- They are the recommended way to represent control characters; in particular, the line feed character 0x0A. They are also the only valid way to specify the control characters #x00 to #x08, #x0B, #x0C, and #x0E to #x1F,which are all illegal in XML 1.0. The most common use case for those illegal characters is within ARINC 661 escape sequences.
- In the GBK and UTF-8 encoding cases: illegal XML characters should only occur within ARINC 661 escape sequences and that is the only context in which backslash-escaping is recommended.

Additional information related to backslash-escaping:

- Backslash-escaped sequences are specific to the XML Definition Files. They are not found in character sequences within the binary DF.
- \\ (two backslashes) can be used to represent a single backslash, in addition to using the other ways provided in this document, namely, \x5C and \.
- There may be implementation dependent contexts (e.g., for string values which could specify MS Windows filenames, which naturally contain backslashes) for which backslash-escaping processing would be suppressed.
- No other backslash specifications (e.g., \n or \t as supported in C/C++) are supported in ARINC 661. Treatment of invalid backslash specification is implementation dependent (e.g., special considerations may be desired to handle conversion of older XML files – see next bullet).
- Older XML DFs (pre-Supplement 4) which did not use backslash-escaping can be converted to the standard format by replacing any single backslash character that they contain with two backslash characters. There may be other conversions required based on how escape sequences were represented in those older XML DF files.
- These backslash-escaping syntaxes are not part of the XML standard, so characters represented in this way are not directly supported by XML Editors or Viewers. Special processing is required in programs that read or write ARINC 661 XML DF files to support backslash-escaping.
- As an example, the following XML yields the string “You are not a banana” with “a banana” having a background color set to “2.”

<prop name=“LabelString”

value=“you are not \x1B\x42\x02a banana”/>

6.2.4.3 Variable-Structure Properties

The datablock, datum, and datapair elements allow saving variable-structure properties (such as the StaticParamBufferExtension’s BufferArray parameter structure) for which it is not practical to assign field names. A **datablock** allows structure-like grouping of data, but without requiring field names.

The number and order of child elements within a datablock is important, because these “fields” do not have names. Nested-structure cases may be handled by

placing a **datablock** within a **datablock** to group the values of the nested structure. As shorthand for a block containing a pair of values such as an (x, y) pair, the **datapair** tag is provided.

Note: In the BufferArray **arrayprop** StaticParamBufferExtension, the number of child elements in each **datablock** immediately under the **arrayprop** must be equal to the BufferFormat's NumberOfFields parameter.

6.3 Document Type Definition (DTD) Specification

6.4 This section deleted in Supplement 6. References

"eXtensible Markup Language (XML)"

A technical recommendation standard of the W3C. W3C Consortium (*www.w3c.org*), release 1.0, February 10, 1998.

"XML Schema"

A technical recommendation standard of the W3C Consortium (*www.w3c.org*), release 1.0, May 2, 2001.

6.5 XML Definition Split with Layer and WidgetSet

To make UA design easier and more manageable, the approach of using Layer and WidgetSet definitions in breaking up UALDs from an XML DF can be considered.

This approach encapsulates parts of UALD into separate collections, i.e., Layer or WidgetSet definitions, where they persist as separate XML files. These collections can be referenced once or multiple times simplifying the creation of UADFs.

The use of such collections is only applicable to ARINC 661 tools and the exchanging of XML files, it does not apply to the exchange between UA and CDS. Therefore, valid UALDs in binary DF (as per Section 2.1.1) must be produced from the XML files before they are used by the CDS, this could be achieved by the expansion of all definitions and references of such collections.

The splitting of an XML DF into multiple XML files has the following benefits:

- **Facilitates modularity and reusability**
- **Increases maintainability with finer change control**
- **Supports collaboration or concurrent development for a UA**

6.5.1 Defining and Referencing

XML DF can be split at the following levels

- **Layer: Encapsulation of a layer definition which can be referenced multiple times by multiple XML DFs.**
- **WidgetSet: Encapsulation of a set of widgets which can be referenced multiple times from a Layer or another WidgetSet.**

For example, the following XML DF

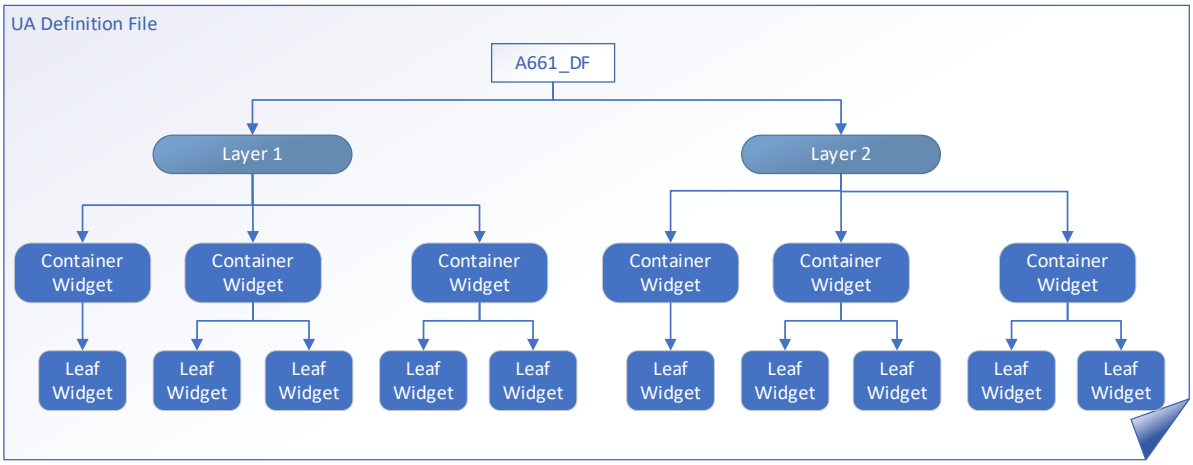


Figure 6.5.1-1 – UADF Using a Single XML DF

could be split as multiple XML files as follows:

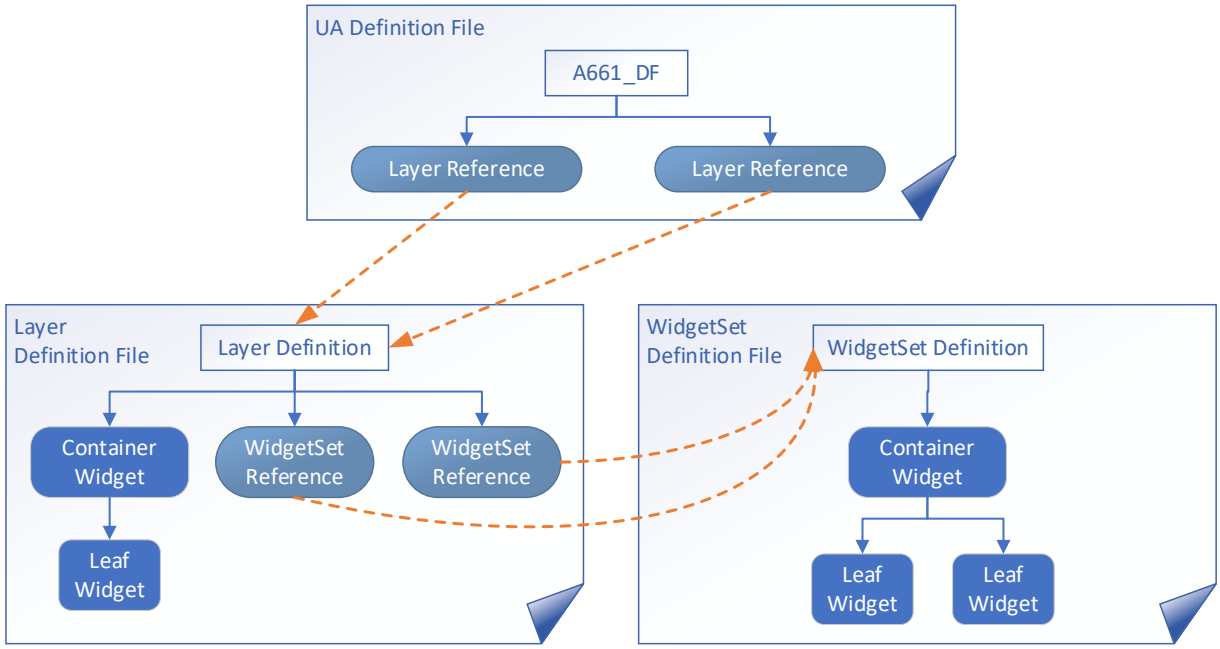


Figure 6.5.1-2 – UADF with Layer and WidgetSets in Separate XML Files

When Layer and WidgetSet references are expanded to produce valid UALDs, the resulting binary DFs from the two XML DF examples above should be the same.

6.5.1.1 Possible Children of Layer and WidgetSet Definitions

Since Layer and WidgetSet definitions are design time artefacts only and expansion is required to produce valid UALDs, the expansion process should ensure the resulting UALDs comply with the Possible Children of Container Widgets rules defined in Section 3.2.3.1. See Section 6.5.2.2.5 for details.

6.5.1.2 Layer

The following two XML elements extend the standard DF Schema

- `<a661_layer_defn>`, root XML element to a Layer Definition XML file. The children are widgets as if they were under the <a661_layer> XML element inside an XML DF.
- `<a661_layer_ref>`, XML element which references a Layer Definition from an XML file.

Layer Definition has the following properties:

Table 6.5.1.2-1 – Layer Definition Properties

Property	Description
DFContext	(optional, string): Specifies the DF which defines the symbol and/or picture tables. The value can be either: *path*: Absolute or relative path of the DF XML file. *type*: Name of the DF which is defined in the name attribute on the <a661_df> xml element.
Height	(optional, ulong): The height of the layer, in 1/100 mm. Present in the XML File only
Width	(optional, ulong): The width of the layer, in 1/100 mm. Present in the XML File only

The following provides an example of Layer Definition XML file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_layer_defn name="Feature01" library_version="0" supp_version="7">
  <model>
    <prop name="DFContext" value="ExampleUA.xml" />
    <prop name="Width" value="0" />
    <prop name="Height" value="0" />
  </model>
  <a661_widget name="Feature01Collection" type="A661_BASIC_CONTAINER">
    ...
    ...
    ...
  </a661_widget>
</a661_layer_defn>
```

Layer Reference has the following properties:

Table 6.5.1.2-2 – Layer Reference Properties

Property	Description
LayerId	(uchar): the layer ID number.
ContextNumber	(ushort): the “context number” of the layer.

Example XML DF using Layer References:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_df name="ExampleUA">
  <model>
    <prop name="ApplicationId" value="1" />
  </model>
  <a661_layer_ref name="Layer01" definition="Feature01.xml">
    <model>
      <prop name="LayerId" value="1" />
      <prop name="ContextNumber" value="0" />
    </model>
  </a661_layer_ref>
  <a661_layer_ref name="Layer02" definition="Feature02.xml">
    <model>
      <prop name="LayerId" value="2" />
      <prop name="ContextNumber" value="0" />
    </model>
  </a661_layer_ref>
  <a661_layer name="Layer03">
    <model>
      <prop name="LayerId" value="3" />
      <prop name="ContextNumber" value="0" />
      <prop name="Width" value="10000" />
      <prop name="Height" value="15000" />
    </model>
    <a661_widget name="DispString" type="A661_LABEL">
      <model>
        <prop name="WidgetIdent" value="1" />
        <prop name="Anonymous" value="A661_TRUE" />
        <prop name="Visible" value="A661_TRUE" />
        <prop name="PosX" value="0" />
        <prop name="PosY" value="4000" />
        <prop name="SizeX" value="6000" />
        <prop name="SizeY" value="1000" />
        <prop name="RotationAngle" value="0" />
        <prop name="StyleSet" value="0" />
        <prop name="MaxStringLength" value="20" />
        <prop name="MotionAllowed" value="A661_TRUE" />
        <prop name="Font" value="0" />
        <prop name="ColorIndex" value="1" />
        <prop name="Alignment" value="A661_CENTER" />
        <prop name="LabelString" value="Standard Layer" />
      </model>
    </a661_widget>
  </a661_layer>
  <a661_layer_ref name="Layer04" definition="Feature01.xml">
    <model>
      <prop name="LayerId" value="4" />
      <prop name="ContextNumber" value="0" />
    </model>
  </a661_layer_ref>
</a661_df>
```

The UADF in this example has four layers – three of which have been implemented using Layer referencing:

- **Feature01 Layer Definition has been referenced twice – Layer01 and Layer04 (without duplication)**
- **Feature02 Layer Definition has been referenced once – Layer02**

A composite UADF XML can include Layer References alongside layers defined in within the DF XML explicitly.

Rules for a Layer Definition are:

- **Any child of a Layer can be a child of a Layer Definition.**

- A Layer Definition XML file can be referenced more than once in a single XML DF using Layer Reference elements.
- A Layer Definition XML file can be referenced in different XML DFs.

6.5.1.3 WidgetSet

The following two XML elements extend the standard DF Schema

- `< a661_widgetset_defn>`, root XML element to a WidgetSet Definition XML file. The children are widgets as if they were under the <a661_layer> XML element inside an XML DF.
- `< a661_widgetset_ref>` XML element which references a WidgetSet Definition XML file.

WidgetSet Definition has the following properties:

Table 6.5.1.3-1 – WidgetSet Definition Properties

Property	Description
DFContext	(optional, string): Specifies the DF which defines the symbol and/or picture tables. The value can be either: *path*: Absolute or relative path of the DF XML file. *type*: Name of the DF which is defined in the name attribute on the <a661_df> xml element.
MinWidgetIdent	(ushort): Lowest widget identifier in the WidgetSet
MaxWidgetIdent	(ushort): Highest widget identifier in the WidgetSet MinWidgetIdent and MaxWidgetIdent together specify the range of widget identifiers in the WidgetSet definition. This is required for managing the widget identifier and avoid/minimize conflicts following WidgetSet updates to ensure that the widget identifiers always remain unique even when the top-level DF is flattened by inlining the WidgetSets.
Height	(optional, ulong): The height of the WidgetSet, in 1/100 mm. Present in the XML file only
Width	(optional, ulong): The width of the WidgetSet, in 1/100 mm. Present in the XML file only

WidgetSet Reference has the following properties:

Table 6.5.1.3-2 – WidgetSet Reference Properties

Property	Description
WidgetIdentOffset	(long): An offset to be applied to all the reference Widgets WidgetIdent.
PosX	(long): Horizontal position of the WidgetSet instance (with respect to the immediate parent).
PosY	(long): Vertical position of the WidgetSet instance (with respect to the immediate parent).

Additionally, a <a661_widgetset_defn> can contain children <a661_widgetset_ref>, for example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_widgetset_defn name="MultipleDials">
  <model>
    <prop name="DFContext" value="ExampleUA.xml" />
    <prop name="MinWidgetIdent" value="0" />
    <prop name="MaxWidgetIdent" value="20" />
    <prop name="Width" value="0" />
    <prop name="Height" value="0" />
  </model>
  <variables_defn>
    <variable name="Size" type="ulong" default="1000"/>
  </variables_defn>
  <a661_widget name="DialScale" type="A661_GP_CROWN">
    <model>
      <prop name="WidgetIdent" value="1" />
      <prop name="Anonymous" value="A661_TRUE" />
      <prop name="Visible" value="A661_TRUE" />
      <prop name="PosX" value="3500" />
      <prop name="PosY" value="3500" />
      <prop name="StartAngle" value="0" />
      <prop name="EndAngle" value="-90" />
      <prop name="InnerRadius" value="1000" />
      <bindprop name="OuterRadius" binding="Size" />
      <prop name="StyleSet" value="0" />
      <prop name="ColorIndex" value="0" />
      <prop name="Filled" value="A661_TRUE" />
      <prop name="FillIndex" value="0" />
      <prop name="Halo" value="A661_FALSE" />
    </model>
  </a661_widget>
</a661_widgetset_defn>
```

6.5.1.3.1 WidgetSet Variables

To aid reusability by tailoring common designs for different purposes, WidgetSet variables provide the mechanism to set parameters in a WidgetSet to values which are specific to each instance. For example, a WidgetSet containing a power indication design could be configured in its instance to display either voltage, current or frequency data.

Use of WidgetSet variable involves the declaration of variables and its bindings to widget parameters within the WidgetSet definition; and the specifying of instance specific values within the WidgetSet reference.

Limitations

Only definition time parameters (D or DR) can bind to a WidgetSet variable, since these variables are only expanded at definition time.

Only simple parameters or structure fields can bind to a WidgetSet variable, for example string, number or enumeration types. Complex types such as arrays and structures cannot bind to widget set variables, as these are complex types.

WidgetSet variable cannot be used for the following special parameters:

- WidgetType
- WidgetIdent

Variable Declaration

The name and default value for each WidgetSet variable are declared within WidgetSet definitions. When resolving variable bindings, the default value is used for the bounded parameters in the expanded instance if instance specific value for the variable is not provided by the WidgetSet reference.

Optionally, the data type for the variables can also be specified to support correct usage of variable bindings.

The following XML fragment shows an example of variable names default values being declared inside a WidgetSet definition.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_widgetset_defn name="DialDefn" library_version="0" supp_version="7">
  <model>
    <prop name="DFContext" value="ExampleUA.xml" />
    <prop name="MinWidgetIdent" value="1" />
    <prop name="MaxWidgetIdent" value="5" />
    <prop name="Width" value="0" />
    <prop name="Height" value="0" />
  </model>
  <variables_defn>
    <variable name="Size" type="ulong" default="1000"/>
    <variable name="Colour" type="uchar" default="10"/>
    <variable name="MyVisible" type="uchar" default="A661_TRUE"/>
  </variables_defn>
…
</a661_widgetset_defn>
```

Variable Binding

For each widget parameter within a WidgetSet definition, instead of using the value attribute, variable binding can be specified by using the binding attribute to reference a variable name.

The following XML fragment demonstrates binding of GpCrown fill and stroke color to a WidgetSet variable named “Colour” and OuterRadius to a variable named “Size”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_widgetset_defn name="DialDefn" library_version="0" supp_version="7">
…
  <a661_widget name="DialScale" type="A661_GP_CROWN">
    <model>
      <prop name="WidgetIdent" value="1" />
      <prop name="Anonymous" value="A661_TRUE" />
      <bindprop name="Visible" binding="MyVisible" />
      <prop name="PosX" value="3500" />
      <prop name="PosY" value="3500" />
      <prop name="StartAngle" value="0" />
      <prop name="EndAngle" value="-90" />
      <prop name="InnerRadius" value="1000" />
      <bindprop name="OuterRadius" binding="Size" />
      <prop name="StyleSet" value="0" />
      <bindprop name="ColorIndex" binding="Colour" />
      <prop name="Filled" value="A661_TRUE" />
      <bindprop name="FillIndex" binding="Colour" />
      <prop name="Halo" value="A661_FALSE" />
    </model>
  </a661_widget>
  <a661_widget name="String" type="A661_LABEL">
    <model>
      <prop name="WidgetIdent" value="2" />
      <prop name="Anonymous" value="A661_TRUE" />
      <prop name="Visible" value="A661_TRUE" />
      <prop name="PosX" value="2500" />
      <prop name="PosY" value="3000" />
      <prop name="SizeX" value="2000" />
```

```
      <prop name="SizeY" value="1000" />
      <prop name="RotationAngle" value="0" />
      <prop name="StyleSet" value="0" />
      <prop name="MaxStringLength" value="20" />
      <prop name="MotionAllowed" value="A661_TRUE" />
      <prop name="Font" value="0" />
      <bindprop name="ColorIndex" binding="Colour" />
      <prop name="Alignment" value="A661_CENTER" />
      <prop name="LabelString" value="$100" />
    </model>
  </a661_widget>
</a661_widgetset_defn>
```

Instance Values

When the WidgetSet is referenced, instance specific values can be assigned against the WidgetSet variables overriding any default value declared in the WidgetSet definition.

The following XML fragment provides an example for setting variables for each WidgetSet instance.

```
<a661_widgetset_ref name="DialInstance01" definition="DialDefn.xml">
  <model>
       <prop name="WidgetIdentOffset" value="20" />
       <prop name="PosX" value="5484" />
       <prop name="PosY" value="6328" />
  </model>
  <variables>
       <prop name="Size" value="2000" />
       <prop name="Colour" value="10" />
       <prop name="MyVisible" value="A661_TRUE" />
  </variables>
</a661_widgetset_ref>
<a661_widgetset_ref name="DialInstance02" definition="DialDefn.xml">
  <model>
       <prop name="WidgetIdentOffset" value="30" />
       <prop name="PosX" value="1423" />
       <prop name="PosY" value="6143" />
  </model>
  <variables>
       <prop name="Size" value="1500" />
       <prop name="Colour" value="1" />
       <bindprop name="MyVisible" binding="EnclosingWSVisible" />
  </variables>
</a661_widgetset_ref>
```

6.5.1.3.2 Widget References

One of the things to consider when using WidgetSets is the ability to reference widgets inside a WidgetSet from outside the WidgetSet. This would be useful when using BufferFormat for grouping widget properties to reduce overhead, or to specify focus navigation order from outside the WidgetSet to within.

Some widgets make references to other widget instances (e.g., BufferFormat) therefore a means to reference widgets within a WidgetSet is needed when specifying design time parameter values. Widget references may use WidgetIdent numbers, Widget names and/or GUIDs:

- Widget references stored as a number should reflect the offset WidgetIdent; i.e., the sum of the WidgetSet references' WidgetIdentOffset and the inner widget's WidgetIdent.

- Widget references using widget names should use a concatenation of the widget set name, '.' and the inner widget name, e.g., "WidgetSetName.InnerWidgetName".
- Widget references using GUIDs should use a similar pattern to widget name references, i.e., <WidgetSet GUID> '.' <InnerWidget GUID>.

A combination of the above may also be used, where inconsistent specification allows user to choose method of resolving the issue. Multiple reference methods may be separated by whitespace and/or a semi-colon.

For example:

"123; <WidgetSetName>.<InnerWidgetName>; <WidgetSetGUID>.<InnerWidgetGUID>"

Example for nested WidgetSets:

"123; <WidgetSetName1>.<WidgetSetName2>.<InnerWidgetName>;
<WidgetSet1GUID>.<WidgetSet2GUID>.<InnerWidgetGUID>"

Note: Names and GUIDs are more resilient to changes within a WidgetSet definition than numeric references. When a name reference is broken, the intended target name is preserved in human readable format; the GUID may be used to resolve a renamed widget.

6.5.1.3.3 Focus Navigation Using Variables

Widgetset definition can define the logical focus navigation order for the children widgets which should be common for all instances. However, the next widget in the focus navigation sequence (after the last child widget in the sequence inside the WidgetSet) should be specified using an interface-level variable "NextFocusedWidget" for each instance.

Consider the following example:

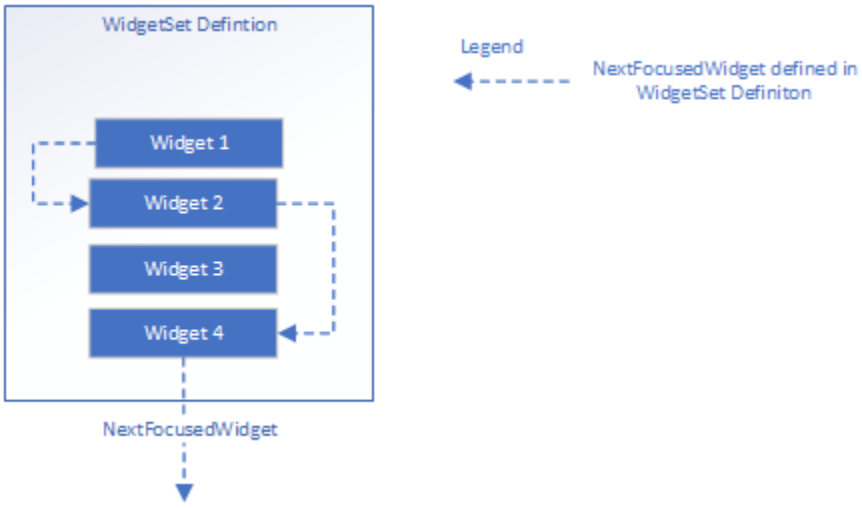


Figure 6.5.1.3.3-1 Focus Example with WidgetSets

This WidgetSet definition in Figure 6.5.1.3.3-1 specifies the internal focus navigation order as follows:

Widget 1 → Widget 2 → Widget 4 → NextFocusedWidget

The variable NextFocusedWidget is used to specify the next widget in the navigation sequence after the focus moves from the WidgetSet instance and should be specified per instance.

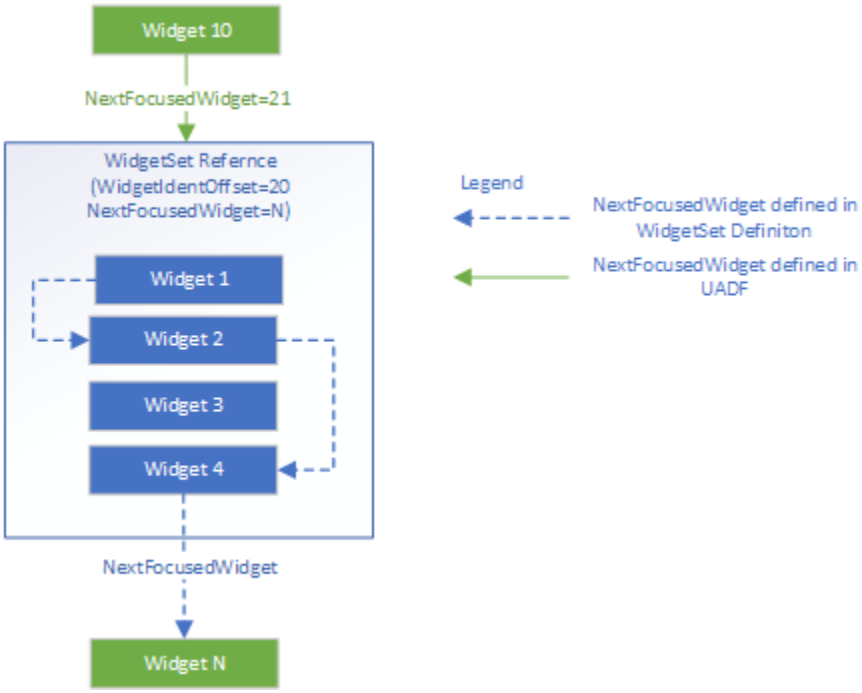


Figure 6.5.1.3.3-2 Example WidgetSet Definition (Focus Navigation)

Figure 6.5.1.3.3-2 shows focus navigation scenario when using WidgetSets. Widget 10 specifies next widget in the focus navigation sequence as Widget 21.

Focus navigation internal to the WidgetSet is then as specified in the WidgetSet definition. NextFocusedWidget variable binding is used to resolve the next widget in the sequence once the focus lands on the last WidgetSet child widget (Widget 4 in the example).

6.5.2 Reference Expansion

When Layer and WidgetSet references go through expansion the following rules should be followed.

6.5.2.1 Layer Reference

All Layer references `<a661_layer_ref>` are replaced with layer definitions `<a661_layer>`, with the contents of their associated Layer definition placed inside the layer.

The new layer definition `<a661_layer>` will inherit the following property values from the from the Layer reference `<a661_layer_ref>`,

- Name
- LayerId
- ContextNumber

6.5.2.2 WidgetSet Reference

All WidgetSet references `<a661_widgetset_ref>` are replaced with the contents of its associated WidgetSet definition.

For each of the widgets defined in the `<a661_widgetset_def>` the following aspects should be considered during expansion.

6.5.2.2.1 Widget Name

Widget Name should be prefix with the <a661_widgetset_ref> Name.

6.5.2.2.2 WidgetIdent

All WidgetIdent parameters of widgets within a WidgetSet should be adjusted by the WidgetSet instance's WidgetIdentOffset defined in the <a661_widgetset_ref>. If there is nesting of WidgetSets, the effect of WidgetIdentOffset is compounded, i.e., offset of the outer WidgetSet needs to be added to the offset to all the inner WidgetSets.

Similarly, widget parameters which reference WidgetIdents (i.e., NextFocsedWidget, ActiveTabbedPanelID) should be adjusted by the WidgetIdentOffset if the referenced widget is within a WidgetSet.

Note that widgets within a WidgetSet may, through the use of WidgetSet variables, refer to a widget outside the WidgetSet definition. Unlike references to widgets within the same WidgetSet, references bound to a variable should not be offset.

6.5.2.2.3 WidgetSet Variables

All widget parameters which reference WidgetSet variables should be replaced with the value defined in the Variables defined in the <a661_widgetset_ref>.

6.5.2.2.4 Widget Positions

The PosX and PosY values of a `<a661_widgetset_ref>` represent the offset for that instance of WidgetSet. Therefore, parameter values related to position for the expanded widgets from the top-level of the WidgetSet definition should account for these offsets, the resulting effect of the offset should be the same as placing the WidgetSet content inside a BasicContainer with the same translation.

In Figure 6.5.2.2.4-1, widgets which need to have the translation applied to are shown in green.

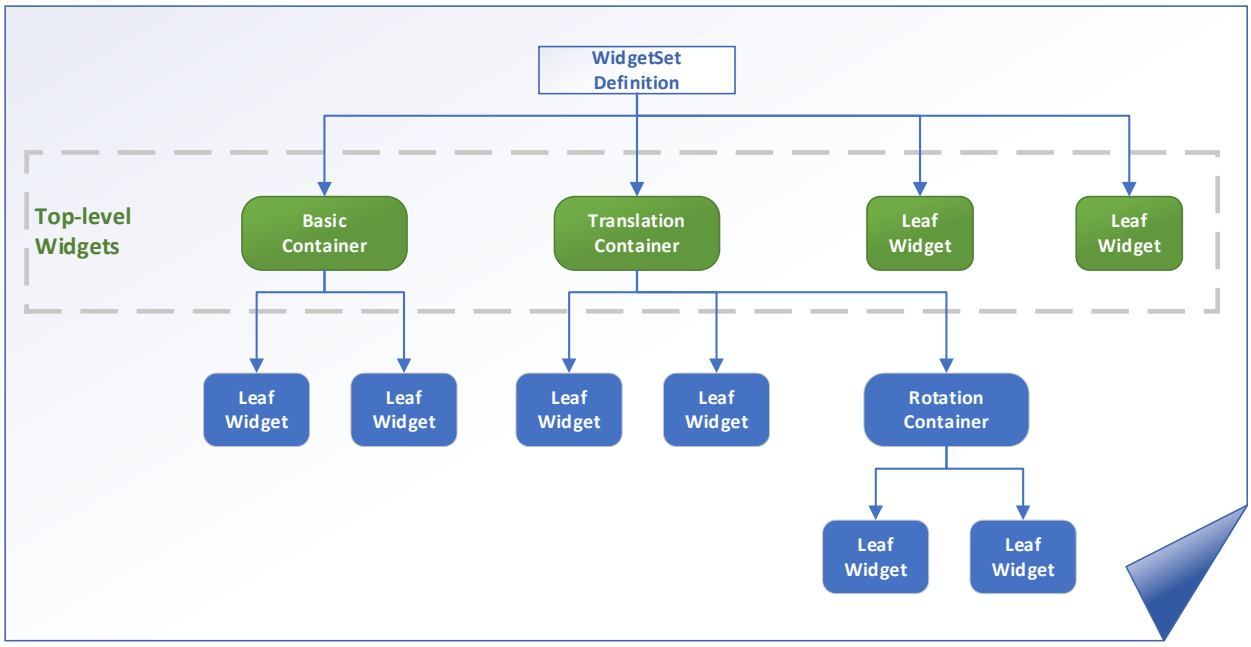


Figure 6.5.2.2.4-1 Widgets Affected by WidgetSet Translation

Note: For the purpose of WidgetSet reference expansion, a graphical widget is considered to be a widget with graphical representation or a container that may have descendant widget(s) with graphical representation.

If a top-level widget is a graphical widget and does not define an origin, i.e., cannot be translated by modifying the definition time parameters, the use of non-zero PosX and PosY for the WidgetSet reference is implementation dependent. e.g., Using the following widgets at the top-level of the WidgetSet would present a problem when translating the WidgetSet reference:

- RotationContainer
- ScaleContainer
- BlinkingContainer
- RefreshContainer
- WatchdogContainer
- NoServiceMonitor
- MapHorz_Source
- MapVert_Source

6.5.2.2.5 Respecting Possible Children of Container Widget Rules

A WidgetSet may only be used within a parent widget if all the WidgetSet's top-level widgets are allowed as children of the parent, such that the expanded layer definition satisfies the Possible Children of Container Widgets (Table 3.2.3.1).

As an exception to this rule, a WidgetSet may contain non-graphical widgets that are only allowed as top-level widgets of a layer. Such widgets must be at the top-level of the WidgetSet. These widgets include:

- BufferFormat
- DataScalingLong
- DataScalingULong
- DataScalingFR180
- BroadcastReceiver
- AnimationGroup
- AnimationOnParam
- AnimationRotation
- AnimationScale
- AnimationTranslation
- EventHandler

In order to comply with Possible Children of Container Widgets rules, the expansion process must insert these non-graphical widgets directly under the layer (instead of where the WidgetSet reference is located), immediately following the top most widget that is an ancestor of their WidgetSet reference. This rule preserves the relative ordering with other such widgets.

Example Layer definition:

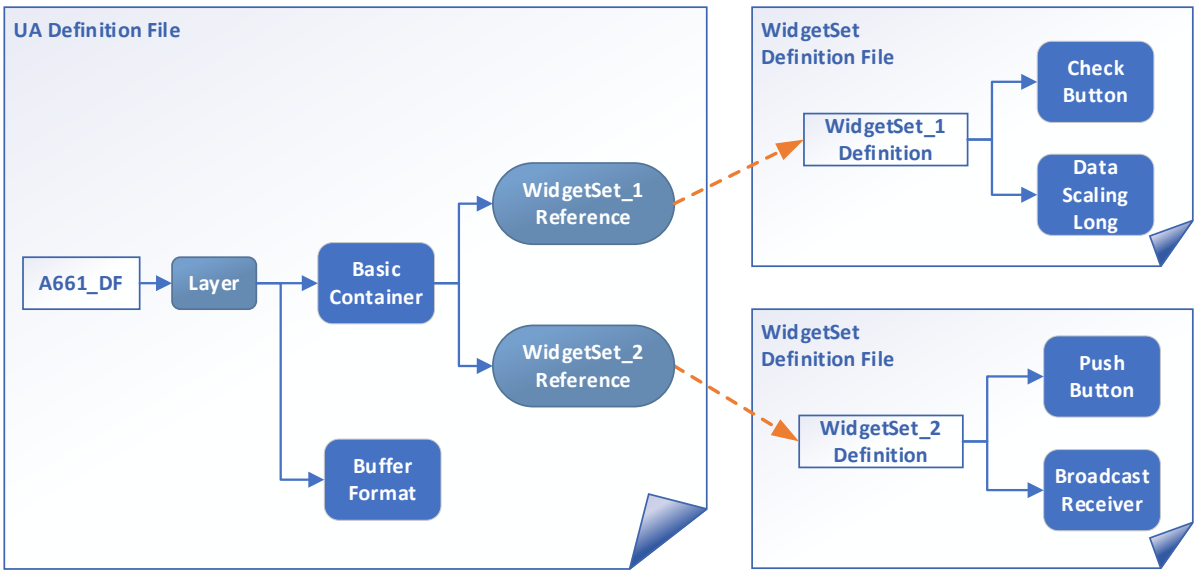


Figure 6.5.2.2.5-1 XML DF with WidgetSet Reference Before Expansion

Example Layer, expanded with non-graphical widgets re-parented:

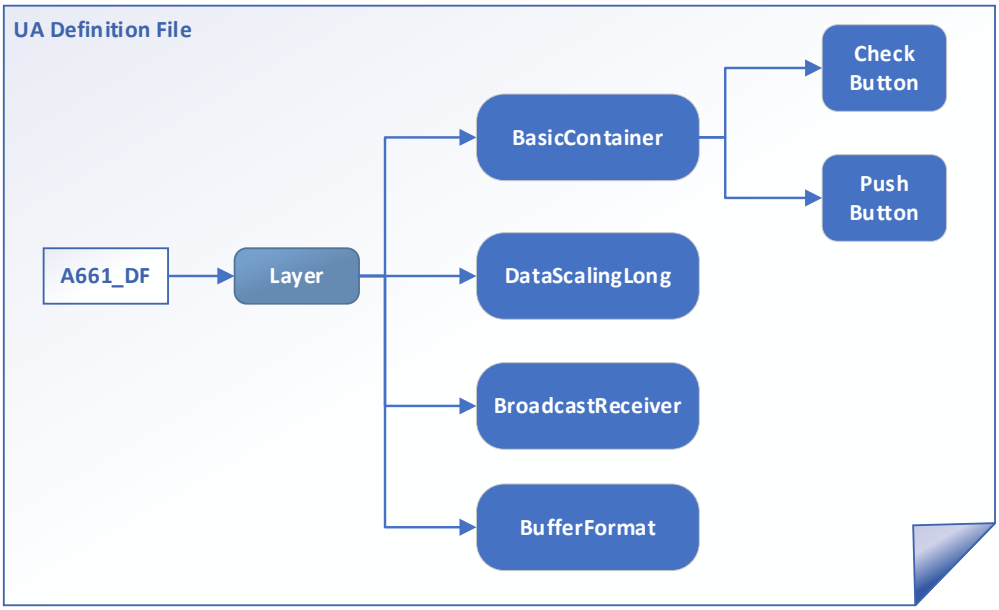


Figure 6.5.2.2.5-2 XML DF After WidgetSet Expansion

6.5.2.2.6 Nesting of WidgetSets

A `<a661_widgetset_defn>` can contain children `<a661_widgetset_ref>`. When the `<a661_widgetset_ref>` is expanded all the rules applied above should be recursively applied.

Therefore, variables can be bound from the parent `<a661_widgetset_defn>` to the child `<a661_widgetset_ref>`. For example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<a661_widgetset_defn name="MultipleDials">
  <model>
    <prop name="DFContext" value="ExampleUA.xml" />
    <prop name="MinWidgetIdent" value="0" />
    <prop name="MaxWidgetIdent" value="20" />
    <prop name="Width" value="0" />
    <prop name="Height" value="0" />
  </model>
  <variables_defn>
    <variable name="Size" type="ulong" default="1000"/>
  </variables_defn>
  <a661_widgetset_ref name="Dial01" definition="DialDefn.xml">
    <model>
      <prop name="WidgetIdentOffset" value="10" />
      <prop name="PosX" value="0" />
      <prop name="PosY" value="0" />
    </model>
    <variables>
      <bindprop name="Size" binding="Size" />
      <prop name="Colour" value="10" />
      <prop name="MyVisible" value="A661_TRUE" />
    </variables>
  </a661_widgetset_ref>
  <a661_widgetset_ref name="Dial02" definition="DialDefn.xml">
    <model>
      <prop name="WidgetIdentOffset" value="20" />
      <prop name="PosX" value="0" />
      <prop name="PosY" value="0" />
    </model>
```

6.0 DEFINITION FILE SPECIFICATION

```
    <variables>
      <bindprop name="Size" binding="Size" />
      <prop name="Colour" value="10" />
      <prop name="MyVisible" value="A661_TRUE" />
    </variables>
  </a661_widgetset_ref>
</a661_widgetset_defn>
```

7.0 PICTURE GRAPHICAL DEFINITION

7.1 Introduction

This section defines how bitmap images can be defined in Definition Files. These bitmap images are identified by a PictureReference number, and they can be referenced by a variety of widgets, such as Picture and PicturePushButton.

Similar to vector symbols defined by the Symbol Graphical Definition language (see Section 5.0 of this specification), bitmaps defined in a Definition File can be referenced by any layer whose User Application Layer Definition (UALD) is defined in the same Definition File as the symbol it wants to reference.

The following scheme should be used to find the appropriate picture for a given PictureReference number:

- If a Picture with matching ID is found in the DF, use it
- Otherwise, use the predefined (global) Picture with that ID

7.2 Picture Definition Structures

Pictures are defined in a Picture structure, which in turn is part of a PictureBlock structure. Each PictureBlock determines the pixel format and color table (if applicable) for all Pictures that it contains.

Table 7-1 below defines the PictureBlock structure. Zero or more PictureBlock structures may appear inside a Definition File between the Definition File header and the User Application Layer Definitions (UALDs), as described in Section 4.5.3.2.

Table 7-1 – Picture Block Structure Exchanged Between UA and CDS at Definition Time

A661_Picture_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_PICTURE_BLOCK	uchar	8	Start keyword opening a picture.
PixelFormat	uchar	8	Defines Pixel and storage format, e.g., A661_PIX_FMT_RGBA_8
NumberOfPicturesInBlock	uchar	8	Number of pictures defined in this block
NumberOfColorTableEntries	uchar	8	Size (number of colors) of the color table. 0 means global CDS color table is referenced, only used with PixelFormats that reference a color table
ColorTableFormat	uchar	8	Defines format of each color table entry, for example A661_PIX_FMT_RGBA_8. Ignored if NumberOfColorTableEntries is zero, only used with PixelFormats that reference a color table.
UnusedPad	N/A	24	0
[Color_Table]		{32}	Color table entries (if defined), padded to align with 4 byte boundary, only used with PixelFormats that reference a color table
{A661_Picture_Structure_DT}+	N/A	{32}+	One or more pictures in this block
Reserved for future use	N/A	32	
A661_END_PICTURE_BLOCK	uchar	8	Keyword ending a block of picture information.
UnusedPad	N/A	24	0

Pictures within these PictureBlocks are defined as described in Table 7-2.

Table 7-2 – PictureStructure

A661_Picture_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_PICTURE	uchar	8	Start keyword opening a picture.
UnusedPad	N/A	8	0
PictureReference	ushort	16	
NumberOfPixelsWidth	ushort	16	Width of the picture in pixels
NumberOfPixelsHeight	ushort	16	Height of the picture in pixels
{ A661_Pixel_Values }+	N/A	{32}+	Information for each pixel, padded to align with 4 byte boundary after the last pixel
A661_END_PICTURE	uchar	8	Keyword ending a picture structure.
UnusedPad	N/A	24	0

In this structure, PictureReference is a numeric identifier that widgets such as Picture and PicturePushButton use to identify a bitmap they wish to display. Pictures are rectangular; the size of a picture is defined by the width and height, measured in the number of pixels used in the bitmap that defines the picture. Note that the details of scaling or mapping this image to pixels of the actual display is dependent on the CDS implementation.

Pixels data ({ A661_Pixel_Values }+) is stored as a sequence of pixels, starting with the top left pixel and then progressing left to right, one line at a time until the bottom line is completed.

The definition of pixel and storage formats is addressed by an enumerated type, which determines both the pixel format and the storage format. While the support for various pixel and storage formats is largely dependent on the CDS implementation, three widely used formats are defined in this specification. CDS implementers may use these formats at their discretion or add additional formats as required.

Table 7-3 – Pixel and Storage Formats

Enumeration Type Value	Pixel Format	Storage Format (bits)
A661_PIX_FMT_RGBA_8	Red, Green, Blue, Alpha	8+8+8+8=32
A661_PIX_FMT_LUMINANCE_ALPHA_8	Luminance, Alpha	8+8=16
A661_PIX_FMT_COLOR_INDEXED_8	Color Index	8

Red, Green, and Blue information in the formats shown in Table 7-3 are stored in 8 bits each, where zero means lowest intensity and 255 means highest intensity of each base color. For Alpha, zero means fully transparent and 255 means fully opaque. Likewise, a luminance of zero has the lowest intensity, and 255 represents the highest intensity.

7.3 Color Tables

Pictures may reference a color table, in which case indices into this table are used instead of explicit RBG values. There are two different scenarios:

1. NumberOfColorTableEntries = 0:

   No color table is defined in the PictureBlock. Instead, color index values in the picture are interpreted the same way as ColorIndex parameters in other widgets, such as GpLine. This means that the global CDS color table is used for the picture. The ColorTableFormat parameter is ignored.
2. NumberOfColorTableEntries > 0:

In this case, the NumberOfColorTableEntries parameter defines how many colors indices are available. The range of color index values is from zero to (NumberOfColorTableEntries – 1), meaning that at most a PictureBlock can define 255 colors in a color table. The ColorTableFormat parameter determines the format of the data describing each color table entry, starting with color index zero.

8.0 WIDGET EXTENSIONS

8.1 General Description

Widget Extensions are a method for adding new optional features to an existing ARINC 661 widget without breaking compatibility with earlier implementations not using the new features. This is done by adding an extension block to the Definition File following the widget instance to be extended. The extension block contains all the parameters necessary to implement the new feature. These parameters may also be runtime modifiable. Multiple extensions may be applied to a single widget, each one applying a new independent feature to the existing widget. Widget Extensions can also cause new events to be sent to the User Application.

8.2 Rules for Extensions

The following rules should be observed for all Widget Extensions.

1. A Widget Extension should always be optional.
2. A Widget Extension should represent a single function.
3. A Widget Extension should add the same feature to all widgets it applies to.
4. A Widget Extension should be able to be applied independently of any other Widget Extension.

The first rule above ensures that a Widget Extension does not cause a widget to become incompatible with its previous definition. One of the major advantages of Widget Extensions is that they preserve backwards compatibility with User Applications that previously used the widget. The second and third rules allow the Extension to be abstracted. The final rule ensures that the Extension is complete.

8.3 Extensions Summary

Table 8.3 summarizes the Extensions.

Table 8.3 – Extensions Summary

Extension Type	Description
EXTENSIONS ADDED FOR SUPPLEMENT 5	
DirectionalTabbingExtension	The DirectionalTabbingExtension allows the Definition File to contain a widget focus navigation order that can be changed at runtime.
CursorEventsExtension	The CursorEventsExtension is an extension which enables a widget to send additional cursor events.
LegendStringAlignmentExtension	The LegendStringAlignmentExtension enables independent alignment (justification) of LegendStrings.
VisibleChildIndexExtension	The VisibleChildIndexExtension allows the UA to interface with the extended widget using child index values as opposed to WidgetId.
InitialFocusExtension	The InitialFocusExtension allows the UA to define the widget as the place where the focus order begins for a format.
FocusStopExtension	The FocusStopExtension allows the UA to define the widget as a place where the focus order stops during a transition of focus across a format.

Extension Type	Description
CursorShapeExtension	This extension allows an application to specify a different cursor shape to be used when the widget is highlighted or has a highlighted item.
EXTENSIONS ADDED FOR SUPPLEMENT 6	
PictureExtension	The PictureExtension allows the addition of one or more pictures to a widget, alongside the corresponding label.
SymbolExtension	The SymbolExtension allows the addition of one or more symbols to a widget, alongside the corresponding label.
TicsArrayExtension	The TicsArrayExtension allows to specify an array of Tics for an interactive widget which manage Tics.
StyleSetExtension	This extension allows an additional StyleSet capacity for a widget for implementations that require more than the 16 bits provided by the native StyleSet parameter.
StaticParamBufferExtension	This extension stores a static (definition-time) array of BufferFormatData values within the CDS. The UA can apply one of these parameter buffers by setting the BufferIndex parameter instead of sending the BufferFormatData parameter data over the network.
MultiSelectionExtension	The MultiSelectionExtension allows the operator to select multiple entries for a widget.
ExcludedRegionsExtension	The ExcludedRegionsExtension defines a list of regions which exclude parkable map items. Map items that are located inside one of these excluded regions are impacted in an implementation dependent manner.
BoundaryCheckExtension	The BoundaryCheckExtension provides a mechanism for a UA to indicate how boundary checking results should be reported.
EXTENSIONS ADDED FOR SUPPLEMENT 7	
ReOrderExtension	The ReOrderExtension allows to change the order of a widget at runtime, relative to its siblings.
WordWrapExtension	The WordWrapExtension allows the UA to indicate that CDS should insert new lines to ensure a string fits within its horizontal size boundary to perform a “word wrap” feature.
CursorEntryEventsExtension	The CursorEntryEventsExtension is an extension which enables a widget which has several entries (such as ScrollList or Tree) to send additional cursor events, and specify on which entry the event occurred
MapHorzCoordSystemEventExtension	The MapHorzCoordSystemEventExtension is an extension which allow the CDS to change and report the range, orientation, and Projection Reference Point on a MapHorz widget based on user input
MapVertCoordSystemEventExtension	The MapVertCoordSystemEventExtension is an extension which allow the CDS to change and report the range and Projection Reference Point on a MapVert widget based on user input

8.4 Possible Widgets for Extension

This section lists all Extensions. Table 8.4 is a cross-reference showing to which widgets each Extension can be applied.

Table 8.4 – Possible Widgets for Extension

Widgets \ Extensions	DirectionalTabbingExtension	CursorEventsExtension	LegendStringAlignmentExtension	VisibleChildIndexExtension	InitialFocusExtension	FocusStopExtension	CursorShapeExtension	PictureExtension	SymbolExtension	TicsArrayExtension	StyleSetExtension	StaticParamBufferExtension	MultiSelectionExtension	ExcludedRegionsExtension	BoundaryCheckExtension	ReorderExtension	WordWrapExtension	CursorEntryEventsExtension	MapHorzCoordSystemEventExtension	MapVertCoordSystemEventExtension
ActiveArea	X	X			X	X	X				X					X				
BasicContainer																X				
BlinkingContainer																X				
BufferFormat												X								
CheckButton	X				X	X	X	X	X		X					X	X			
ComboBox	X				X	X	X	X	X		X					X	X	X		
Connector																X				
CursorPosOverlay																X				
EditBoxMasked	X				X	X	X				X					X				
EditBoxNumeric	X		X		X	X	X			X	X					X				
EditBoxText	X				X	X	X				X					X				
GpArcCircle											X					X				
GpArcEllipse											X					X				
GpCrown											X					X				
GpLine											X					X				
GpLinePolar											X					X				
GpRectangle											X					X				
GpTriangle											X					X				
Label											X					X	X			
LabelComplex											X					X	X			
MapHorz																X			X	
MapHorz_ItemList							X								X	X				
MapHorz_Source																X			X	
MaskContainer																X				
Panel											X					X				
Picture											X					X				
PicturePushButton	X	X			X	X	X				X					X	X			
PictureToggleButton	X				X	X	X				X					X	X			
PopUpMenu									X		X					X				
PopUpMenuButton	X				X	X	X		X		X					X	X			

8.0 WIDGET EXTENSIONS

Widgets \ Extensions	DirectionalTabbingExtension	CursorEventsExtension	LegendStringAlignmentExtension	VisibleChildIndexExtension	InitialFocusExtension	FocusStopExtension	CursorShapeExtension	PictureExtension	SymbolExtension	TicsArrayExtension	StyleSetExtension	StaticParamBufferExtension	MultiSelectionExtension	ExcludedRegionsExtension	BoundaryCheckExtension	ReorderExtension	WordWrapExtension	CursorEntryEventsExtension	MapHorzCoordSystemEventExtension	MapVertCoordSystemEventExtension
PopUpPanel											X					X	X			
PushButton	X	X			X	X	X	X	X		X					X	X			
RadioBox																X				
RotationContainer																X				
ScrollList	X				X	X	X	X	X		X		X			X	X	X		
ScrollPanel											X					X				
Symbol											X					X				
TabbedPanel	X				X	X	X		X		X					X	X			
TabbedPanelGroup				X							X					X				
ToggleButton	X				X	X	X	X	X		X					X	X			
TranslationContainer																X				
WIDGETS ADDED FOR SUPPLEMENT 1																				
ComboBoxEdit	X				X	X	X	X	X		X					X	X			
EditBoxMultiLine	X				X	X	X				X					X				
ExternalSource											X					X				
MapGrid																X				
MapVert																X				X
MapVert_ItemList																X				
MapVert_Source																X				X
MenuBar																X				
WIDGETS ADDED FOR SUPPLEMENT 2																				
MutuallyExclusiveContainer				X												X				
PictureAnimated											X					X				
ProxyButton																				
SelectionListButton	X				X	X	X	X	X		X					X	X			
Slider	X				X	X	X				X					X				
SymbolAnimated											X					X				
WatchdogContainer																				
WIDGETS ADDED FOR SUPPLEMENT 3																				
CursorRef																X				
CursorOver							X				X					X				
EditBoxNumericBCD	X		X		X	X	X				X					X				
FocusIn																				
FocusLink																				
FocusOut																				

8.0 WIDGET EXTENSIONS

Widgets \ Extensions	DirectionalTabbingExtension	CursorEventsExtension	LegendStringAlignmentExtension	VisibleChildIndexExtension	InitialFocusExtension	FocusStopExtension	CursorShapeExtension	PictureExtension	SymbolExtension	TicsArrayExtension	StyleSetExtension	StaticParamBufferExtension	MultiSelectionExtension	ExcludedRegionsExtension	BoundaryCheckExtension	ReorderExtension	WordWrapExtension	CursorEntryEventsExtension	MapHorzCoordSystemEventExtension	MapVertCoordSystemEventExtension
ShuffleToFitContainer											X					X				
SizeToFitContainer											X					X				
WIDGETS ADDED FOR SUPPLEMENT 4																				
PopUpPanelButton	X				X	X	X		X		X					X	X			
SymbolPushButton	X	X			X	X	X				X					X				
SymbolToggleButton	X				X	X	X				X					X				
WIDGETS ADDED FOR SUPPLEMENT 5																				
Broadcast Receiver																				
DataScalingLong																				
DataScalingULong																				
DataScalingFR180																				
GpPolyline											X					X				
MapHorz_Container																X				
MapHorz_Panel											X					X				
NoServiceMonitor																				
NumericReadout			X								X					X	X			
PagingContainer				X							X					X				
WIDGETS ADDED FOR SUPPLEMENT 6																				
AnimationGroup																				
AnimationOnParam																				
AnimationRotation																				
AnimationScale																				
AnimationTranslation																				
DataConnector																X				
EventHandler																				
FramingRefreshContainer																				
GestureArea											X					X				
InkArea					X	X	X				X					X				
KeyboardArea							X									X				
MapBoundary														X						
MapHorz_VertexBuffer																				
MapVert_Container																X				
MapVert_Panel																X				
MultiStateButton	X				X	X	X	X	X		X					X				
ScrollWheelArea							X									X				
TouchArea											X					X				

Widgets \ Extensions	DirectionalTabbingExtension	CursorEventsExtension	LegendStringAlignmentExtension	VisibleChildIndexExtension	InitialFocusExtension	FocusStopExtension	CursorShapeExtension	PictureExtension	SymbolExtension	TicsArrayExtension	StyleSetExtension	StaticParamBufferExtension	MultiSelectionExtension	ExcludedRegionsExtension	BoundaryCheckExtension	ReorderExtension	WordWrapExtension	CursorEntryEventsExtension	MapHorzCoordSystemEventExtension	MapVertCoordSystemEventExtension
WIDGETS ADDED FOR SUPPLEMENT 7																				
MapExternalSource																			X	X
ScaleContainer																X				
Selector	X				X	X	X	X	X		X					X		X		
Tree	X				X	X	X	X	X		X		X			X	X	X		

8.5 Widget Extensions Added for Supplement 5

This section describes the characteristics of each ARINC 661 Widget Extension. For each Extension, the following are provided:

Extension Name

- Extension Description
- Functional feature or capability added by the Extension
- List of widgets the Extension can be applied to
- Definition File structure of the Extension
- List of parameters added to the widget
- Creation Structure of the Extension
- Runtime Modifiable Parameters (if applicable)
- Structure of all RMP parameters added to the widget
- Events (if applicable)
- Structure of all Events added to the widget

8.5.1 DirectionalTabbingExtension

Description:
The DirectionalTabbingExtension allows the Definition File to contain a widget focus navigation order that can be changed at runtime. The CDS uses the navigation order provided by this extension to move the widget focus from one widget to another in response to crew member inputs made via tabbers, joysticks, and similar devices. This order is in addition to any other order defined in the DF (i.e., via the NextFocusedWidget parameter), but does not necessarily supersede that order.

To include a widget instance in this navigation sequence, a DirectionalTabbingExtension is added to the DF. The parameter

DirectionWidgetIdArray specifies the set of all other widgets immediately reachable from the widget instance. This array also indicates in which general direction each reachable widget lies. One WidgetIdent is specified for each possible direction. The number of directions is specified by the NumDirections parameter.

The maximum number of directions is implementation dependent.

The CDS determines to which widget the focus should move based on the number of possible directions and the order the WidgetIdents appear in the TabDirectionWidgetIdArray. The details of this are implementation dependent.

DirectionalTabbingExtension Parameters are defined in Table 8.5.1-1.

Table 8.5.1-1 – DirectionalTabbingExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_DIRECTIONAL_TABBING_EXTENSION
NumDirections	D	Defines the number of directions for which TabDirectionWidgetId are specified in the Definition File.
TabDirectionWidgetIdArray	DR	Array of TabDirectionWidgetIdents, contains one WidgetIdent in each direction. A value of zero (0) means no widget is specified for that direction.

DirectionalTabbingExtension Creation Structure is defined in Table 8.5.1-2.

Table 8.5.1-2 – DirectionalTabbingExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_DIRECTIONAL_TABBING_EXTENSION
NumDirections	uchar	8	Must be less than the OEM defined maximum
UnusedPad	N/A	8	0
TabDirectionWidgetIdArray [NumDirections]	{ushort}+	16 * Num Directions + Pad	There are NumDirections TabDirectionWidgetIds. The complete array is followed by 2 bytes padding if NumDirections is an odd number.

Available SetParameter identifiers and associated data structure are defined in Table 8.5.1-3.

Table 8.5.1-3 – DirectionalTabbingExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
TabDirectionWidgetIdArray	NA	{32}+	A661_TAB_DIRECTION_WIDGET_ID_ARRAY

8.5.2 CursorEventsExtension

Description:
The CursorEventsExtension is an extension which enables a widget to send additional cursor events. For example, a PushButton only has a Click Cursor event but it may be desired to monitor the duration of such a click. The addition of a Press event and a Release event through this extension would meet this need.

The UA can choose (via the EventIDArray parameter) which events it wants to receive among the available events added by this extension. A CDS can implement some, or all, of the list of events described below. The list may be expanded in future Supplements to this standard.

To avoid corruptions, the CDS should send one event at a time, and the UA should take the events into account one at a time, per the order in which they were received.

Example: if a UA were only interested in the Double-Click event, it should set the NumEvents parameter to one (1) and the EventIdArray parameter to the enumeration value A661_CURSOR_DOUBLE_CLICKED.

Note: How widget events are translated into "selection" for widgets is implementation dependent.

AdditionalCursorEvents Parameters are defined in Table 8.5.2-1.

Table 8.5.2-1 – CursorEventsExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_CURSOR_EVENTS_EXTENSION
NumEvents	D	The number of events in the following EventsArray
EventIdArray	D	Array to define which events the UA wants to receive using the enumerations below. A661_CURSOR_PRESSED A661_CURSOR_RELEASED A661_CURSOR_CLICKED A661_CURSOR_DOUBLE_CLICKED A661_CURSOR_RIGHT_CLICKED

AdditionalCursorEvents Creation Structure is defined in Table 8.5.2-2.

Table 8.5.2-2 – CursorEventsExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_CURSOR_EVENTS_EXTENSION
NumEvents	ushort	16	
EventIdArray[NumEvents]	{ushort}*	16 * NumEvents + Pad	There are NumEvents events in the array. The complete array is followed by 2 bytes padding if NumEvents is an odd number.

Table 8.4.2-3 defines the specific event sent by the CursorEventsExtension to the owner application.

Table 8.5.2-3 – CursorEventsExtension Event Structure: A661_EVT_CURSOR_EVENT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_EVENT
EventId	ushort	16	Enumeration indicating which event occurred

8.5.3 LegendStringAlignmentExtension

Description:
The LegendStringAlignmentExtension enables independent alignment (justification) of the LegendString for EditBoxNumeric, EditBoxNumericBCD, and NumericReadout widgets.

This extension adds a widget parameter which controls the LegendString's alignment relative to its starting position within the widget display edit box area. CDS will use this new parameter to align the LegendString independently of the numeric value.

When using this extension, the UA must ensure that the MaxLegendStringLength and LegendAreaSizeX widget parameters are set correctly so that during alignment the Legend String does not go beyond the borders of the widget display area.

This extension does not add any new events.

COMMENTARY

EditBoxNumeric, EditBoxNumericBCD, and NumericReadout combine a numeric value to be displayed with a text string represented by the LegendString parameter. Without this extension, the combined string can be aligned within the widget display area as a unit. However, if the contents or format of the numeric value change, an undesirable change in the LegendString's position can result. Implementers use padding and other workarounds to prevent this effect, but these are tedious and increase development cost.

LegendStringAlignment Parameters are defined in Table 8.5.3-1.

Table 8.5.3-1 – LegendStringAlignment Parameters

Parameters	Change	Description
ExtensionType	D	A661_LEGEND_ALIGN_EXTENSION
LegendAlignment	DR	Justification of the LegendString within the widget display area, relative to its starting point, which is controlled by the LegendPosition parameter (see Sections 3.3.10 and 3.6.1). CENTER LEFT RIGHT

LegendStringAlignment Creation Structure is defined in Table 8.5.3-2.

Table 8.5.3-2 – LegendStringAlignment Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_LEGEND_ALIGN_EXTENSION
LegendAlignment	uchar	8	A661_CENTER A661_LEFT A661_RIGHT
UnusedPad	N/A	8	0

Table 8.5.3-3 – LegendStringAlignment Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure
LegendAlignment	uchar	8	A661_LEGEND_ALIGNMENT

8.5.4 VisibleChildIndexExtension

Description:
The VisibleChildIndexExtension allows the UA to interface with the extended widget using child index values as opposed to WidgetId.

Child indices are defined by the order in which the widgets appear in the definition file (i.e., the first widget in definition file that references a particular parent is considered the 1st child, in other words the one corresponding to index 1. The next is considered the 2nd child and so on). In some cases, referencing children in this way is more convenient for the UA developer.

Note: The runtime parameter and event below are in addition to the existing runtime parameters and events that communicate the child's WidgetId on the extended widget.
The VisibleChildIndex event is sent only when these existing events are sent.

At definition time the VisibleChildIndex parameter will take priority over the widget ID based parameter. At runtime the last parameter received will be used.

VisibleChildIndexExtension Parameters are defined in Table 8.5.4-1.

Table 8.5.4-1 – VisibleChildIndexExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_CHILD_INDEX_EXTENSION
VisibleChildIndex	DR	The index of the visible child. A value of 0 means that no widget is visible.

VisibleChildIndexExtension Creation Structure is defined in Table 8.5.4-2.

Table 8.5.4-2 – VisibleChildIndexExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_CHILD_INDEX_EXTENSION
VisibleChildIndex	ushort	16	

Available SetParameter identifiers and associated data structure are defined in Table 8.5.4-3.

Table 8.5.4-3 – VisibleChildIndexExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
VisibleChildIndex	ushort	16	A661_VISIBLE_CHILD_INDEX

VisibleChildIndexExtension Event Structures: A661_EVT_VISIBLE_CHILD_INDEX are defined in Table 8.5.4-4.

Table 8.5.4-4 – VisibleChildIndexExtension Event Structures

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_VISIBLE_CHILD_INDEX
VisibleChildIndex	ushort	16	

8.5.5 InitialFocusExtension

Description:
The InitialFocusExtension allows the UA to define the widget as the place where the focus order begins for a format. The InitialFocus parameter can be used to define which widget should first have focus when the focus motion device is actuated.

As described in Section 3.1.3.5, UAs may define a navigation order for focus motion through the use of the NextFocusedWidget and AutomaticFocusMotion parameters. Focus motion may be controlled by a device such as an arrow key, tabber knob, scroll wheel, etc.

The InitialFocus parameter defines a default focus position if none is currently available. The UA designer should define only one widget with the InitialFocus

parameter set to True. If more than one InitialFocus parameter is True for the widgets on a display, the behavior is implementation dependent.

InitialFocusExtension Parameters are defined in Table 8.5.5-1.

Table 8.5.5-1 – InitialFocusExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_INITIAL_FOCUS_EXTENSION
InitialFocus	DR	Indicates that this widget should be the default start point for focus control when a window is first displayed.

InitialFocusExtension Creation Structure is defined in Table 8.5.5-2.

Table 8.5.5-2 – InitialFocusExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_INITIAL_FOCUS_EXTENSION
UnusedPad	uchar	8	0
InitialFocus	uchar	8	A661_FALSE A661_TRUE

Available SetParameter identifiers and associated data structure are defined in Table 8.5.5-3.

Table 8.5.5-3 – InitialFocusExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
InitialFocus	uchar	8	A661_INITIAL_FOCUS

8.5.6 FocusStopExtension

Description:
The FocusStopExtension allows the UA to define the widget as a place where the focus order stops during a transition of focus across a format.

As described in Section 3.1.3.5, UAs may define a navigation order for focus motion through the use of the NextFocusedWidget and AutomaticFocusMotion parameters. Focus motion may be controlled by a device such as an arrow key, tabber knob, scroll wheel, etc. Since some of these devices may allow the rapid transition of focus across widgets, it may be desirable to define certain widgets at which the focus should stop.

For example, suppose a wrap-around tabber sequence is defined and the UA would like to ensure that the focus stops at the last button on a page. If a focus motion device is “spun” such that the focus moves across many widgets, then it could be difficult to ensure that the focus stops at the last button. With the FocusStopExtension, the StopForward parameter can be used to identify where the focus should stop when the motion control device is moving the focus in the forward direction. In this example, the last button on the page would have its StopForward parameter set to True to ensure that the focus stopped at that widget when the motion control device was spun.

Similarly, the StopBackward parameter indicates where the focus should stop when the motion control device is moving the focus in the backward direction.

The use of the term “stop” in this extension description does not imply the end of the focus navigation order. When the focus “stops” at a widget with StopForward or StopBackward set to True, the next focus motion device movement still continues

along the focus sequence defined with the NextFocusedMotion parameters. Therefore, it is more akin to a pause or delay in which the focus stays on the associated widget until further movement along the focus sequence is initiated.

FocusStopExtension Parameters are defined in Table 8.5.6-1.

Table 8.5.6-1 – FocusStopExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_FOCUS_STOP_EXTENSION
StopForward	DR	Indicates whether forward focus control should stop on this widget.
StopBackward	DR	Indicates whether backward focus control should stop on this widget.

FocusStopExtension Creation Structure is defined in Table 8.5.6-2.

Table 8.5.6-2 – FocusStopExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_FOCUS_STOP_EXTENSION
StopForward	uchar	8	A661_FALSE A661_TRUE
StopBackward	uchar	8	A661_FALSE A661_TRUE

Available SetParameter identifiers and associated data structure are defined in Table 8.5.6-3.

Table 8.5.6-3 – FocusStopExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
StopForward	uchar	8	A661_STOP_FORWARD
StopBackward	uchar	8	A661_STOP_BACKWARD

8.5.7 CursorShapeExtension

Description:
The CursorShapeExtension can be attached to interactive widgets, including MapHorz_ItemList. This extension allows an application to specify a different cursor shape be used when the widget is highlighted or has a highlighted item.

When a widget or map item is highlighted, and its EnableChange parameter is set to True, then the cursor is changed to the shape indicated by the CursorIndex parameter. When the widget or map item is no longer highlighted, or if the EnableChange parameter is changed to False, the cursor is changed back to its previous shape.

The CursorIndex values are implementation dependent.

CursorShapeExtension Parameters are defined in Table 8.5.7-1.

Table 8.5.7-1 – CursorShapeExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_CURSOR_SHAPE_EXTENSION
AllowChange	DR	Indicates whether the cursor shape should change when the widget or a map item (in the case of MapHorz_ItemList) is highlighted.
CursorIndex	DR	Indicates which cursor shape should be used when the widget or a map item (in the case of MapHorz_ItemList) is highlighted.

CursorShapeExtension Creation Structure is defined in Table 8.5.7-2.

Table 8.5.7-2 – CursorShapeExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_CURSOR_SHAPE_EXTENSION
AllowChange	uchar	8	A661_FALSE A661_TRUE
CursorIndex	uchar	8	

Available SetParameter identifiers and associated data structure are defined in Table 8.5.7-3.

Table 8.5.7-3 – CursorShapeExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
AllowChange	uchar	8	A661_ALLOW_CHANGE
CursorIndex	uchar	8	A661_CURSOR_INDEX

8.6 Widget Extensions Added for Supplement 6

8.6.1 PictureExtension

Description:

The PictureExtension allows the addition of one or more pictures to a widget, alongside the corresponding label.

Note: when this extension is used on a PushButton or ToggleButton, its usage is very similar to the PicturePushButton or PictureToggleButton.

The array of pictures defines the pictures used for the widget:

- For widgets which have only one possible label (such as A661_PUSH_BUTTON), only the first index of the picture array is significant, the other ones will be ignored by the CDS.
- For widgets which have two possible labels (i.e., A661_TOGGLE_BUTTON), only the first two indices of the picture array are significant, the other ones will be ignored by the CDS. The first index in the picture array is associated with the first label in the creation structure.
- For widgets which have an array of labels (such as A661_COMBO_BOX), the index of the picture corresponds to the index of the corresponding label. If there are more pictures than labels, the remaining pictures will be ignored (except for case described below). If there are more labels than pictures, the remaining labels will not have an associated picture.

- **For widgets which have an array of labels and a specific label (such as A661_SELECTION_LIST_BUTTON), the index of the picture associated to the specific label should correspond to the last label array index plus one.**

PictureExtension Parameters are defined in Table 8.6.1-1.

Table 8.6.1-1 – PictureExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_PICTURE_EXTENSION
PicturePosition	DR	Position of the picture with respect to the label within the widget CENTER LEFT RIGHT TOP BOTTOM
MaxNumberOfPictures	D	The maximum number of picture references to store
NumberOfPictures	DR	The number of picture references to store
PictureArray	DR	Array of pictures references stored in the CDS. The definition order set the sequence order.

PictureExtension Creation Structure is defined in Table 8.6.1-2.

Table 8.6.1-2 – PictureExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_PICTURE_EXTENSION
MaxNumberOfPictures	ushort	16	
NumberOfPictures	ushort	16	
PicturePosition	uchar	8	A661_LEFT A661_CENTER A661_RIGHT A661_TOP A661_BOTTOM
UnusedPad	N/A	8	0
PictureArray	{ushort}+	16* NumberOf Pictures + PAD	Array of pictures references Followed by zero or two extra NULL for alignment on 32 bits.

Table 8.6.1-3 – PictureExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NumberOfPictures	ushort	16	A661_NUMBER_OF_PICTURES
PictureArray [up to MaxNumberOfPictures]	N/A	{32}+	A661_PICTURE_ENTRY_ARRAY
PicturePosition	**uchar**	**8**	**A661_PICTURE_POSITION**

8.6.2 SymbolExtension

Description:

The SymbolExtension allows the addition of one or more symbols to a widget, alongside the corresponding label.

Note: When this extension is used on a PushButton or ToggleButton, its usage is very similar to the SymbolPushButton or SymbolToggleButton.

The array of symbols defines the symbols used for the widget :

- For widgets which have only one possible label (such as A661_PUSH_BUTTON), only the first index of the symbol array is significant, the other ones will be ignored by the CDS.
- For widgets which have two possible labels (i.e., A661_TOGGLE_BUTTON), only the first two indices of the symbol array are significant, the other ones will be ignored by the CDS. The first index in the symbol array is associated with the first label in the creation structure.
- For widgets which have an array of labels (such as A661_COMBO_BOX), the index of the symbol corresponds to the index of the corresponding label. If there are more symbols than labels, the remaining symbols will be ignored **(except for case described below)**. If there are more labels than symbols, the remaining labels will not have an associated symbol.
- **For widgets which have an array of labels and a specific label (such as A661_SELECTION_LIST_BUTTON), the index of the symbol associated to the specific label should correspond to the last label array index plus one.**

SymbolExtension Parameters are defined in Table 8.6.2-1.

Table 8.6.2-1– SymbolExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_SYMBOL_EXTENSION
SymbolPosition	D**R**	Position of the symbol with respect to the label within the widget CENTER LEFT RIGHT TOP BOTTOM
MaxNumberOfSymbols	D	The maximum number of symbol references to store
NumberOfSymbols	DR	The number of symbol references to store
SymbolArray	DR	Array of symbol references stored in the CDS. The definition order set the sequence order.

SymbolExtension Creation Structure is defined in Table 8.6.2-2.

Table 8.6.2-2 – SymbolExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_SYMBOL_EXTENSION
MaxNumberOfSymbols	ushort	16	
NumberOfSymbols	ushort	16	
SymbolPosition	uchar	8	A661_LEFT A661_CENTER A661_RIGHT A661_TOP A661_BOTTOM
UnusedPad	N/A	8	0
SymbolArray	{ushort}+	16 * NumberOf Symbols + PAD	Array of symbol references Followed by zero or two extra NULL for alignment on 32 bits.

Table 8.6.2-3 – SymbolExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NumberOfSymbols	ushort	16	A661_NUMBER_OF_SYMBOLS
SymbolArray [up to MaxNumberOfSymbols]	N/A	{32}+	A661_SYMBOL_ENTRY_ARRAY
SymbolPosition	uchar	8	A661_SYMBOL_POSITION

8.6.3 TicsArrayExtension

Description:
The TicsArrayExtension allows to specify an array of Tics for a Numeric Edit box which manage Tics.

The TicsCoarseArray and TicsFineArray are used to specify contiguous range of values for using the Tics:

Example TicsArray:

TicsFineArray[1].Tics	Tics before TicsFineArray[1].ValueMax
TicsFineArray[1].ValueMax	
TicsFineArray[2].Tics	Tics between TicsFineArray[1].ValueMax and TicsFineArray[2].ValueMax
TicsFineArray[2].ValueMax	
…	
TicsFineArray[i-1].ValueMax	
TicsFineArray[i].Tics	Tics between TicsFineArray[i-1].ValueMax and TicsFineArray[i].ValueMax
TicsFineArray[i].ValueMax	
…	
TicsFineArray[NumberOfTics].ValueMax	
TicsFineArray[NumberOfTics].Tics	Tics between TicsFineArray[NumberOfTics-1]. ValueMax and TicsFineArray[NumberOfTics].ValueMax

Algorithm (example with TicsFine):

- If the value is smaller than TicsFineArray[1].ValueMax, then the associated Tic will be TicsFineArray[1].Tics.
- If the value is between TicsFineArray[i-1].ValueMax and TicsFineArray[i].ValueMax, then the associated Tic will be TicsFineArray[i].Tics.
- If the value is greater than TicsFineArray[NumberOfTics].ValueMax then the associated Tic will be TicsFineArray[NumberOfTics].Tics.
- If a Tic is equal to 0, then the original Tic parameter for the widget will be used.

Note that the MaxNumberOfTicsCoarse and NumberOfTicsCoarse, and the MaxNumberOfTicsFine and NumberOfTicsFine, represent the maximum or current dimension of the associated array, not the number of Tics. Two values are defined for each range except the last one:

- The Tic to use between the previous maximum value and the next maximum value
- The maximum value to use the Tic

Example: Consider a TicsFineArray for this extension used on an EditBoxNumeric widget which allows to modify the BINGO (low fuel value):

(500, 10000)	**Tics before 10000 = 500**
(2000, 20000)	Tics between 10000 and 20000 = 2000

So, in this case, if the current value is 2500, each TicFine will increase (or decrease) the value by 500 (3000, 3500, etc.). After the value has been increased to 10000, then each TicFine would increase the value by 2000 (12000, 14000, etc.). After the value has been increased to 20000, each TicFine would still increase the value by 2000.

TicsArrayExtension Parameters are defined in Table 8.6.3-1.

Table 8.6.3-1 – TicsArrayExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_TICS_ARRAY_EXTENSION
MaxNumberOfTicsCoarse	D	Maximum number of elements for the TicsCoarseArray
NumberOfTicsCoarse	DR	Number of elements for the TicsCoarseArray
TicsCoarseArray	DR	The array of TicsCoarse as (Tic, Value) pairs
MaxNumberOfTicsFine	D	Maximum number of elements for the TicsFineArray
NumberOfTicsFine	DR	Number of elements for the TicsFineArray
TicsFineArray	DR	The array of TicsFine as (Tic, Value) pairs

TicsArrayExtension Creation Structure is defined in Table 8.6.3-2.

Table 8.6.3-2 – TicsArrayExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_TICS_ARRAY_EXTENSION
UnusedPad	N/A	16	0
MaxNumberOfTicsCoarse	ushort	16	
NumberOfTicsCoarse	ushort	16	
MaxNumberOfTicsFine	ushort	16	
NumberOfTicsFine	ushort	16	
TicsCoarseArray	{float, float}+	64 * NumberOfTicsCoarse	
TicsFineArray	{float, float}+	64 * NumberOfTicsFine	

Available SetParameter identifiers and associated data structure are defined in Table 8.6.3-3.

Table 8.6.3-3 – TicsArrayExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
NumberOfTicsCoarse	ushort	16	A661_NUMBER_OF_TICS_COARSE
NumberOfTicsFine	ushort	16	A661_NUMBER_OF_TICS_FINE
TicsCoarseArray [up to MaxNumberOfTicsCoarse]	{float x 2}+	{64}+	A661_TICS_COARSE_ARRAY
TicsFineArray [up to MaxNumberOfTicsFine]	{float x 2}+	{64}+	A661_TICS_FINE_ARRAY

8.6.4 StyleSetExtension

Description:

The StyleSetExtension can be attached to most graphical widgets, including widgets that already have a StyleSet parameter. This extension allows an additional StyleSet capacity for a widget for implementations that require more than the 16 bits provided by the native StyleSet parameter.

Additional 16-bit StyleSet fields are provided in an array, and each array element is individually addressable at run-time. The parameter MaximumArraySize dictates the maximum number of additional 16-bit StyleSet fields. The values used in the StyleSetArray are implementation dependent.

All widgets of the same type should have the same MaxArraySize (for example, all instances of a PushButton widget should have equivalent MaxArraySize values), otherwise behavior is CDS-dependent.

The StyleSetExtension should be used in the same manner as the native StyleSet parameter. The existence of a StyleSetExtension does not supersede or replace the native StyleSet parameter where it exists, rather it offers additional fields to be used in addition to the native StyleSet.

StyleSetExtension Parameters are defined in Table 8.6.4-1.

Table 8.6.4-1 – StyleSetExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_STYLE_SET_EXTENSION
MaxStyleSetArraySize	D	
StyleSetArraySize	DR	
StyleSetArray	DR	Additional implementation dependent StyleSet characteristics.

StyleSetExtension Creation Structure is defined in Table 8.6.4-2.

Table 8.6.4-2 – StyleSetExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_STYLE_SET_EXTENSION
MaxStyleSetArraySize	uchar	8	
StyleSetArraySize	uchar	8	A661_STYLE_SET_ARRAY_SIZE
StyleSetArray	{ushort}	16 * MaxArraySize	There are MaxArraySize StyleSet elements, padded to align with 32-bit boundary.

Available SetParameter identifiers and associated data structure are defined in Table 8.6.4-3.

Table 8.6.4-3 – StyleSetExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
StyleSetArray	{ushort}	{32}+	A661_STYLE_SET_ARRAY
StyleSetArraySize	uchar	8	A661_STYLE_SET_ARRAY_SIZE

8.6.5 StaticParamBufferExtension

Description:

This extension stores a static (definition-time) array of BufferFormatData values within the CDS. The UA can apply one of these parameter buffers by setting the BufferIndex parameter instead of sending the BufferFormatData parameter data over the network.

COMMENTARY:

By using this extension to enumerate and store sets of widget parameter values on the CDS, UA developers may:

- Reduce network traffic upon layer-activation by storing initial parameter values on the CDS, and
- Avoid duplicating mutually-exclusive widget trees where presentation-specific parameter values (layout, language, color, etc.) change between copies, or
- Avoid network traffic if BufferFormats are already used to switch between presentation aspects.

Possible Widgets for Extension:

BufferFormat

StaticParamBufferExtension Parameters are defined in Table 8.6.5-1.

Table 8.6.5-1 – StaticParamBufferExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_STATIC_PARAM_BUFFER_EXTENSION
NumberOfBuffers	D	Number of buffers which can be applied to the BufferFormat at runtime.
BufferArray	D	Array of buffers, where each entry contains values corresponding to each pair of WidgetIdent/ParameterIdent defined in the attached widget.
BufferIndex	R	Index of BufferArray whose data to apply.

StaticParamBufferExtension Creation Structure is defined in Table 8.6.5.2.

Table 8.6.5-2 – StaticParamBufferExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_STATIC_PARAM_BUFFER_EXTENSION
NumberOfBuffers	ushort	16	Number of entries in BufferArray.
BufferArray	{StaticParamBuffer}+	{32}+	Array of StaticParamBuffer structures.

Table 8.6.5-3 defines the StaticParamBuffer structure.

Refer also to BufferFormat widget description in Section 3.3.4 and the BufferFormat data alignment and padding info in Section 3.3.4.1.

Table 8.6.5-3 – StaticParamBuffer Structure

A661_StaticParamBuffer_Structure	Type	Size (bits)	Description
ParamBufferSize	ushort	16	Size of BufferFormatData, in bytes.
UnusedPad	N/A	16	
{BufferFormatData}+	N/A	{32}+	Parameter Buffer (see Section 3.3.4, BufferFormat)

Available SetParameter identifiers and associated data structure are defined in Table 8.6.5-4.

Table 8.6.5-4 – StaticParamBufferExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
BufferIndex	ushort	16	A661_PARAM_BUFFER_INDEX

8.6.6 MultiSelectionExtension

Description:
The MultiSelectionExtension allows the operator to select multiple entries for a widget.

The CDS may allow the operator to perform a selection of multiple entries in a list for widgets that include this extension. In case a multiple selection operation is performed by the operator, the CDS sends the A661_EVT_MULT_SELECTION event to describe the current state of the selection (instead of A661_EVT_SEL_ENTRY_CHANGE which is sent for single item selections). The indices refer to entries in the entry array which the CDS considers selected.

The UA may modify parameters to set or clear the selected state of entries. The SelectedEntryArray can be sent to choose a set of entries for selection by specifying a list of affected entries and the new selection state for each entry. The SelectAll

parameter can be sent to cause all entries to be selected. The UnselectAll parameter can be sent to clear the selection of all entries.

Note that the SelectedEntryArray parameter and the A661_EVT_MULT_SELECTION event are in terms of CDS indexing and not UA indexing if applicable (i.e., Scroll List).

MultiSelectionExtension Parameters are defined in Table 8.6.6-1.

Table 8.6.6-1 – MultiSelectionExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_MULTI_SELECTION_EXTENSION
MaximumSelectableEntries	D	The number of selectable items allowed
SelectAll	R	Selects all entries
UnselectAll	R	Unselect all entries
SelectedEntryArray	R	List of currently selected entries (each item is an index into the entry array and an indication of whether the entry is selected or not)

MultiSelectionExtension Creation Structure is defined in Table 8.6.6-2.

Table 8.6.6-2 – MultiSelectionExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_MULTI_SELECTION_EXTENSION
MaximumSelectableEntries	ushort	16	

Available SetParameter identifiers and associated data structure are defined in Table 8.6.6-3.

Table 8.6.6-3 – MultiSelectionExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
SelectAll	N/A	8	A661_SELECT_ALL
UnselectAll	N/A	8	A661_UNSELECT_ALL
SelectedEntryArray	N/A	{32}+	A661_SELECTED_ENTRY_ARRAY

MultiSelectionExtension Event Structures: A661_EVT_MULTI_SELECTION are defined in Table 8.6.6-4.

Table 8.6.6-4 – MultiSelectionExtension Event Structures: A661_EVT_MULT_SELECTION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MULT_SELECTION
NumEntries	ushort	16	Number of currently selected entries
EntryIndices	N/A	{32}+	Array of entry indices for currently selected entries (plus pad if an odd number)

8.6.7 ExcludedRegionsExtension

Description:
The ExcludedRegionsExtension defines a list of regions which exclude parkable map items. Map items that are located inside one of these excluded regions are impacted in an implementation dependent manner. For example, they may be parked along the edge of the region or removed from the display altogether.

The excluded regions are based on geometric shapes (e.g., circular arcs, triangles, rectangles). The shapes are all defined in units of hundredths of mm relative to the screen reference point.

ExcludedRegionsExtension Parameters are defined in Table 8.6.7-1.

Table 8.6.7-1 – ExcludedRegionsExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_EXCLUDED_REGIONS_EXTENSION
MaxNumberOfRegions	D	Defines the maximum number of excluded regions
ExcludedRegionsList	DR	A list of regions which exclude parkable map items
EffectivityArray	DR	Defines whether each region is effective or not: A661_TRUE A661_FALSE

ExcludedRegionsExtension Creation Structure is defined in Table 8.6.7-2.

Table 8.6.7-2 – ExcludedRegionsExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_EXCLUDED_REGIONS_EXTENSION
MaxNumberOfRegions	uchar	8	
UnusedPad	NA	8	0
EffectivityArray [MaxNumberOfRegions]	{uchar}+	8 * MaxNumber OfRegions	
ExcludedRegionsList [MaxNumberOfRegions]	{ulong}+	Sum of individual region struct size	There are MaxNumberOfRegions ExcludedRegionStructs in the list. See Table 8.5.7-3 for possible ExcludedRegionStruct types.

Available ExcludedRegionStructs are listed in Table 8.6.7-3.

Table 8.6.7-3 – ExcludedRegionStruct Types

StructName	Size (bits)	Description
ExcludedArcCircle	192	Defines a circular arc from which parkable map items are excluded. See Table 8.5.7-4 for the definition of the ExcludedArcCircle structure.
ExcludedTriangle	224	Defines a triangular region from which parkable map items are excluded. See Table 8.5.7-5 for the definition of the ExcludedTriangle structure.
ExcludedRectangle	160	Defines a rectangular region from which parkable map items are excluded. See Table 8.5.7-6 for the definition of the ExcludedRectangle structure.
NotUsed	32	Marks this exclusion region as currently unused. See Table 8.5.7-7 for the definition of the NotUsed structure.

ExcludedArcCircle structure is defined in Table 8.6.7-4.

Table 8.6.7-4 – ExcludedArcCircle Structure

Field Name	Type	Size (bits)	Description
ExcludedItemIndex	uchar	8	
ExcludedRegionType	uchar	8	A661_EXCLUDED_ARC_CIRCLE
UnusedPad	N/A	16	0
PosX	long	32	The center X position of the arc or circle.
PosY	long	32	The center Y position of the arc or circle.
Radius	ulong	32	The radius of the arc or circle.
StartAngle	fr(180)	32	The angle (referenced from the center position) that defines the start of the arc or circle.
EndAngle	fr(180)	32	The angle (referenced from the center position) that defines the end of the arc or circle.

ExcludedTriangle structure is defined in Table 8.6.7-5.

Table 8.6.7-5 – ExcludedTriangle Structure

Field Name	Type	Size (bits)	Description
ExcludedItemIndex	uchar	8	
ExcludedRegionType	uchar	8	A661_EXCLUDED_TRIANGLE
UnusedPad	N/A	16	0
PosX	long	32	The X start position of the triangle.
PosY	long	32	The Y start position of the triangle.
PosX2	long	32	The X position of the second vertex of the triangle
PosY2	long	32	The Y position of the second vertex of the triangle
PosX3	long	32	The X position of the third vertex of the triangle
PosY3	long	32	The Y position of the third vertex of the triangle

ExcludedRectangle structure is defined in Table 8.6.7-6.

Table 8.6.7-6 – ExcludedRectangle Structure

Field Name	Type	Size (bits)	Description
ExcludedItemIndex	uchar	8	
ExcludedRegionType	uchar	8	A661_EXCLUDED_RECTANGLE
UnusedPad	N/A	16	0
PosX	long	32	The X start position of the lower left corner.
PosY	long	32	The Y start position of the lower left corner.
SizeX	ulong	32	The X dimension size (width) of the rectangle.
SizeY	ulong	32	The Y dimension size (height) of the rectangle.

ExcludedNotUsed structure is defined in Table 8.6.7-7.

Table 8.6.7-7 – ExcludedNotUsed Structure

Field Name	Type	Size (bits)	Description
ExcludedItemIndex	uchar	8	
ExcludedRegionType	uchar	8	A661_NOT_USED
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 8.6.7-8.

Table 8.6.7-8 – ExcludedRegionsExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
ExcludedRegionsList	NA	{32}+	A661_EXCLUDED_REGIONS_LIST
EffectivityArray[up to MaxNumberOfRegions]	N/A	{32}+	A661_EFFECTIVITY_ARRAY

8.6.7.1 A661_ParameterStructure_BufferOfExclusionItems

A661_ParameterStructure_BufferOfExclusionItems as used for the ExcludedRegionsExtension is defined in Table 8.6.7-9.

Table 8.6.7-9 – A661_ParameterStructure_BufferOfExclusionItems

A661_ParameterStructure	Size (bits)	Description
ParameterIdent	16	A661_EXCLUDED_REGIONS_LIST
ClearFlag	8	If set, all items will be set to NOT_USED by CDS before setting the specified Items.
NumberofItems	8	Number of items modified by the command
{ItemStructures}+	{32}+	See Table 8.5.7-3 for possible item structures.

8.6.8 BoundaryCheckExtension

Description:
The BoundaryCheckExtension provides a mechanism for a UA to indicate how boundary checking results should be reported. The parameter BoundaryCheckMode defines which type of event should be generated for maps item list widgets that use the Boundary_Check map item: only out-of-bounds events, only in-bounds events, or the minimum size list among the two.

Generation of events is dependent on a MapBoundary widget being defined as an ancestor of the MapHorz_ItemList.

A map item is considered out-of-bounds if it is beyond (outside) the shape defined by the MapBoundary region. This includes any region defined in the MapBoundary's ExclusionRegions extension: a map item within an active exclusion region is considered out-of-bounds. Since there are many different types and shapes of map items, the determination for when a map item is considered outside the bounds of the map widget is implementation dependent.

The events report an array of map item index values that share the same result (either in-bounds or out-of-bounds). The A661_EVT_ITEM_OUT_OF_BOUNDS event includes a list of the map item indices which are all out-of-bounds. Conversely, the A661_EVT_ITEM_IN_BOUNDS event includes a list of the map item indices which are all in-bounds. Any map item not in the list has the opposite boundary check result.

If the BoundaryCheckMode is A661_MINIMUM_SIZE, then whichever event size is smaller between A661_EVT_ITEM_OUT_OF_BOUNDS and A661_EVT_ITEM_IN_BOUNDS is sent. For example, if there are 20 map items subject to the boundary check, and 9 or fewer are out-of-bounds, then A661_EVT_OUT_OF_BOUNDS would be the event sent. If they are the same size,

then the either event type may be used. This limits worst-case size of the array to be half of the map items subject to the boundary check.

The event is sent after a BufferOfMapItems update, or when a change occurs in the boundary check results between BufferOfMapItems updates. Since map updates may be made across multiple BufferOfMapItems parameters, CDS needs to retain whether boundary checking is applied to a set of map items (i.e., the boundary checking state defined for a set of items persists until an update on that set of items is received).

This event applies to the following map item types (note that this is a different list than the map item types which are applied to the MapBoundary widget):

- LEGEND_ANCHOR
- LEGEND_ANCHOR_ROTATED
- SYMBOL_GENERIC
- SYMBOL_GENERIC_INTERACTIVE
- SYMBOL_ROTATED
- SYMBOL_ROTATED_INTERACTIVE
- SYMBOL_RUNWAY
- SYMBOL_RUNWAY_INTERACTIVE
- SYMBOL_TARGET
- SYMBOL_TARGET_INTERACTIVE
- PARKABLE_SYMBOL
- PARKABLE_SYMBOL_INTERACTIVE

BoundaryCheckExtension Parameters are defined in Table 8.6.8-1.

Table 8.6.8-1 – BoundaryCheckExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_BOUNDARY_CHECK_EXTENSION
BoundaryCheckMode	DR	Indicates which type of boundary check event should be generated

The BoundaryCheckExtension Creation Structure is defined in Table 8.6.8-2.

Table 8.6.8-2 – BoundaryCheckExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_BOUNDARY_CHECK_EXTENSION
BoundaryCheckMode	uchar	8	A661_OUT_OF_BOUNDS: send only list of map items in-bounds. A661_IN_BOUNDS: send only list of map items in-bounds. A661_MINIMUM_SIZE: send whichever list is smaller.
UnusedPad	uchar	8	0

Available SetParameter identifiers and associated data structure are defined in Table 8.6.8-3.

Table 8.6.8-3 – BoundaryCheckExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
BoundaryCheckMode	uchar	8	A661_BOUNDARY_CHECK_MODE

BoundaryCheckExtension Event Structures are defined in Tables 8.5.8-4 and 8.5.8-5.

Table 8.6.8-4 – BoundaryCheckExtension Event Structure: A661_EVT_ITEM_OUT_OF_BOUNDS

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ITEM_OUT_OF_BOUNDS
NumberOfItems	ushort	16	Number of map items whose boundary check result are reported in ItemIndexArray.
ItemIndexArray [NumberOfItems]	{ushort}+	{32}+	Array of indices of map items that have been determined to be out-of-bounds, padded to align to 32-bit boundary. Array contents should be sorted in ascending order of map item index.

Table 8.6.8-5 – BoundaryCheckExtension Event Structure: A661_EVT_ITEM_IN_BOUNDS

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_ITEM_IN_BOUNDS
NumberOfItems	ushort	16	Number of map items whose boundary check result are reported in ItemIndexArray.
ItemIndexArray [NumberOfItems]	{ushort}+	{32}+	Array of indices of map items that have been determined to be in-bounds, padded to align to 32-bit boundary. Array contents should be sorted in ascending order of map item index.

8.7 Widget Extensions Added for Supplement 7

8.7.1 ReorderExtension

Description:
The ReorderExtension allows the change the order of a widget at runtime, relative to its siblings. The order runtime change will only be possible on a branch of a widget tree (it is only possible to change the order of a widget relative to its siblings). Two ordering changes are possible:

- **bringing the widget at the front position relative to its siblings,**
- **bringing the widget at the back position relative to its siblings**

The reordering will not only change the graphical order of the widgets but may also change their layout relative to their parent. For example, using this extension on a child of a ShuffleToFit or SizeToFit container will change the layout of the children in their parent container.

Example:

If we have the following Definition File :

- **TranslationContainer**
  - **Label**
  - **GPArcCircle**
  - **GPLinePolar**
- **GPRectangle**
- **GPTriangle**
- **GPLine**

If we receive the ToFront setParameter for the GpArcCircle widget, the order of the tree will become:

- **TranslationContainer**
  - **Label**
  - **GPLinePolar**
  - **GPArcCircle**
- **GPRectangle**
- **GPTriangle**
- **GPLine**

Notes :

1. **Using this extension with a container which support the VisibleChildIndexExtension (such as MutuallyExclusiveContainer) can lead to problems in some cases if a child of this container also has a VisibleChildIndex extension. For example, if the child of a MutuallyExclusiveContainer is reordered in the container, the children indices will change, and the UA must take care of these indices changes if setting the VisibleChildIndex after this reordering**
2. **Using this extension can have an undesired side effect on Gp widgets using Halo parameters in some cases (see Section 3.1.3.3)**
3. **The UA can completely reorder the widgets subtree by sending ToBack or ToFront repeatedly on several widgets in the subtree**

Possible Widgets for Extension:

See Table 8.3.

ReorderExtension Parameters are defined in Table 8.7.1-1.

Table 8.7.1-1 – ReorderExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_REORDER_EXTENSION
Reorder	R	Parameter indicating that the widget should be reordered relative of its siblings in its parent container A661_TO_FRONT A661_TO_BACK

ReorderExtension Creation Structure is defined in Table 8.7.1-2.

Table 8.7.1-2 – ReorderExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_REORDER_EXTENSION
UnusedPad	N/A	16	0

Available SetParameter identifiers and associated data structure are defined in Table 8.7.1-3.

Table 8.7.1-3 – ReorderExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
Reorder	uchar	8	A661_REORDER

8.7.2 WordWrapExtension

Description:
The WordWrapExtension allows the UA to indicate that CDS should insert new lines to ensure a string fits within its horizontal size boundary (e.g., SizeX) to perform a "word wrap" feature. The resulting alignment of the text should be as if the UA had included CR/LF characters within the text string itself. Any CR/LF characters inserted due to word wrapping should be omitted from event fields that send the string value back to the UA, e.g., for Edit Box widget events.

The word wrapping behavior (e.g., whether to impose wrapping only at word boundaries or within words) can be dictated by the WrapStyle parameter:

- If WrapStyle is A661_PRESERVE_WORD, and a whole word cannot fit within the remaining width of the widget on its current line, then the whole word is wrapped to a new line. For example, if the width of the widget resulted in the word "really" extending beyond the width, the entire word "really" could be wrapped to the next line:

  "The 2019 Phoenix Suns really need a point guard" becomes:

  "The 2019 Phoenix Suns
  really need a point guard."

- If WrapStyle is A661_PRESERVE_CHARACTER, and a whole character cannot fit within the remaining widget of the widget on its current line, then the remainder of the word starting with that character is wrapped

to a new line, regardless of if it is in the middle of a word. For example, if the width of the widget resulted in the 'y' in "really" extending beyond the width, the word could be split across two lines as follows:

"The 2019 Phoenix Suns reall
y need a point guard."

- If WrapStyle is A661_HYPHENATE, and a whole word cannot fit within the remaining width of the widget on its current line, then the word is broken by a hyphen, with the portion of the word after the hyphen wrapping to a new line. For instance, one possible method of resolving a case where the width of the widget resulted in the word "really" extending beyond the width, then "really" could be split by a hyphen across two lines as follows:

"The 2019 Phoenix Suns real-
ly need a point guard."

For widgets with multiple text strings, the word wrapping and style apply to all text string parameters and array elements defined for those widgets.

If the first word of a line does not fit within the width when WrapStyle is A661_PRESERVE_WORD, then the result is implementation dependent, but it is recommended that the CDS impose the A661_PRESERVE_CHARACTER or A661_HYPHENATE for the word in that case.

The method(s) by which word and character preservation is handled by the CDS is implementation dependent, including how WrapStyle is applied across various languages. Treatment of white space at the wrapping boundary is implementation dependent.

The relationship between word wrapping and text alignment is implementation dependent, but it is recommended that alignment be applied to each line after word wrapping has been imposed on a string.

Possible Widgets for Extension:

See Table 8.4.

WordWrapExtension Parameters are defined in Table 8.7.2-1.

Table 8.7.2-1 – WordWrapExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_WORD_WRAP_EXTENSION
WordWrap	DR	Indication of whether to apply word wrapping to ensure a string fits within its horizontal size boundary (e.g. SizeX).
WrapStyle	DR	Indicates kind of word wrapping features to apply.

WordWrapExtension Creation Structure is defined in Table 8.7.2-2.

Table 8.7.2-2 – WordWrapExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_WORDWRAP_EXTENSION
WordWrap	uchar	8	A661_FALSE A661_TRUE
WrapStyle	uchar	8	Method of wrapping to apply: A661_PRESERVE_WORD A661_PRESERVE_CHARACTER A661_HYPHENATE Custom enumeration values should use the range 0x80 – 0xFF.

Available SetParameter identifiers and associated data structure are defined in Table 8.7.2-3.

Table 8.7.2-3 – WordWrapExtension Runtime Modifiable Parameters

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure
WordWrap	uchar	8	A661_WORD_WRAP
WrapStyle	uchar	8	A661_WRAP_STYLE

8.7.3 CursorEntryEventsExtension

Description:
The CursorEntryEventsExtension is an extension which enables a widget which has several entries (such as ScrollList or Tree) to send additional cursor events and specify on which entry the event occurred. For example, it may be desirable for an UA to receive an event if the user right click on a tree entry. This extension is similar to the CursorEvents extension.

The UA can choose (via the EventIDArray parameter) which events it wants to receive among the available events added by this extension. A CDS can implement some or all of the list of events described below. The list may be expanded in future Supplements to this standard.

To avoid corruptions, the CDS should send one event at a time, and the UA should take the events into account one at a time, per the order in which they were received.

The following example presents the value of the event sent by the extension if ENTRY 3 has been selected:

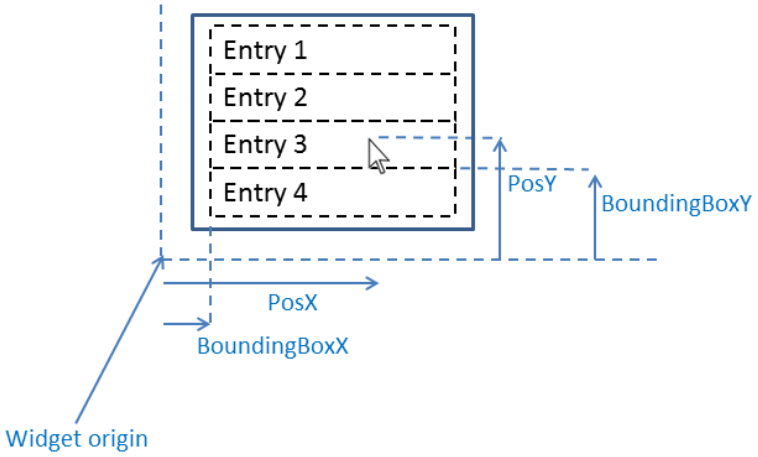


Figure 8.7.3-1 – Usage Example on a ScrollList Widget

Example: if a UA were only interested in the Double-Click event, it should set the NumEvents parameter to one (1) and the EventIdArray parameter to the enumeration value A661_CURSOR_DOUBLE_CLICKED.

Note: How widget events are translated into "selection" for widgets is implementation dependent.

The X and Y positions of the A661_EVT_CURSOR_ENTRY_EVENT can allow the UA to position a Popup at the position of the selected entry.

For example, if we have the following Definition File :

- **ScrollList with CursorEntryEventsExtension with EventIdArray = {A661_CURSOR_RIGHT_CLICKED}**
- **Panel (Visibility set to False at design-time)**
  - **Panel content including buttons to set options for the ScrollList selected element**

On a right-click on the ScrollList received by the UA by the A661_EVT_CURSOR_ENTRY_EVENT event, the Panel Visibility is set to true, and the position of the Panel is set to the PosX and PosY of the event.

Notes:

1. **The CursorEntryEvent sent by the extension returns both the cursor position and the bounding box for the selected entry.**
2. **The BoundingBoxX and BoundingBoxY parameters are not taking into account the widget dimension, which means that these values can be negative (for example if the widget has been scrolled internally).**
3. **Entry selection is performed by the widget on which the extension is applied**

Additional CursorEntryEventsExtension Parameters are defined in Table 8.7.3-1.

Table 8.7.3-1 – CursorEntryEventsExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_CURSOR_ENTRY_EVENTS_EXTENSION
NumEvents	D	The number of events in the following EventsArray
EventIdArray	D	Array to define which events the UA wants to receive using the enumerations below. A661_CURSOR_PRESSED A661_CURSOR_RELEASED A661_CURSOR_CLICKED A661_CURSOR_DOUBLE_CLICKED A661_CURSOR_RIGHT_CLICKED

Additional CursorEntryEventsExtension Creation Structure is defined in Table 8.7.3-2.

Table 8.7.3-2 – CursorEntryEventsExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_CURSOR_ENTRY_EVENTS_EXTENSION
NumEvents	ushort	16	
EventIdArray[NumEvents]	{ushort}*	16 * NumEvents + Pad	There are NumEvents events in the array. The complete array is followed by 2 bytes padding if NumEvents is an odd number.

Table 8.7.3-3 defines the specific event sent by the CursorEntryEventsExtension to the owner application.

Table 8.7.3-3 – CursorEntryEventsExtension Event Structure: A661_EVT_CURSOR_ENTRY_EVENT

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_CURSOR_ENTRY_EVENT
EventId	ushort	16	Enumeration indicating which event occurred
SelectedEntry	ushort	16	Index of the entry on which the event occurred
UnusedPad	N/A	16	
X	long	32	The X position of the cursor relative to the widget
Y	long	32	The Y position of the cursor relative to the widget
BoundingBoxX	long	32	The X position of the bounding box of the entry relative to the widget
BoundingBoxY	long	32	The Y position of the bounding box of the entry relative to the widget
BoundingBoxSizeX	ulong	32	The width of the bounding box of the entry
BoundingBoxSizeY	ulong	32	The height of the bounding box of the entry

8.7.4 MapHorzCoordSystemEventExtension

Description:
The MapHorzCoordSystemEventExtension is an extension which allows the CDS to report changes to the range, orientation, and Projection Reference Point on a MapHorz widget (for example a drag and drop on the Map could trigger a A661_EVT_PROJECTION_REFERENCE event, or a pinch interaction could trigger a A661_EVT_RANGE event).

The UA can choose (via the EventIDArray parameter) which events it wants to receive among the available events added by this extension.

The availability of these events for the CDS, and the way the CDS interprets the user interaction to send the values, are implementation dependent. For example, the event could be sent continuously when the user performs a drag and drop on the map, or only at the end of the gesture.

Note that this extension is available on both the MapHorz, MapHorz_Source, and MapExternalSource widgets. When this extension is applied on a MapHorz_Source or MapExternalSource widget, it sends the events corresponding to user interactions on the MapHorz parent of the MapHorz_Source or MapExternalSource.

MapHorzCoordSystemEvent Parameters are defined in Table 8.7.4-1.

Table 8.7.4-1 – MapHorzCoordSystemEventExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_MAPHORZ_COORD_SYSTEM_EVENT_EXTENSION
NumEvents	D	The number of events in the following EventsArray
EventIdArray	D	Array to define which events the UA wants to receive using the enumerations below. A661_RANGE A661_ORIENTATION A661_PROJECTION_REFERENCE
ReportMode	D	A661_REPORT_ON_CDS Report only changes originating from the CDS A661_REPORT_ALL Report all changes

MapHorzCoordSystemEventExtension Creation Structure is defined in Table 8.7.4-2.

Table 8.7.4-2 – MapHorzCoordSystemEventExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_MAPHORZ_COORD_SYSTEM_EVENT_EXTENSION
NumEvents	ushort	16	
ReportMode	uchar	8	A661_REPORT_ON_CDS A661_REPORT_ALL
UnusedPad	N/A	24	0
EventIdArray[NumEvents]	{ushort}*	16 * NumEvents + Pad	There are NumEvents events in the array. The complete array is followed by 2 bytes padding if NumEvents is an odd number.

A661_EVT_HORZ_RANGE is defined in Table 8.7.4.3

Table 8.7.4.3 – MapHorzCoordSystemEventExtension Event Structures: A661_EVT_MAPHORZ_RANGE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPHORZ_RANGE
UnusedPad	N/A	16	0
Range	fr(32768)	32	Geo-referenced range expressed in nm

A661_EVT_HORZ_ORIENTATION is defined in Table 8.7.4.4

Table 8.7.4.4 – MapHorzCoordinateSystemEventExtension Event Structures: A661_EVT_MAPHORZ_ORIENTATION

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPHORZ_ORIENTATION
UnusedPad	N/A	16	0
Orientation	fr(180)	32	Angle between True North and the up-direction of the display at the PRP

A661_EVT_HORZ_PROJECTION_REFERENCE is defined in Table 8.7.4.5

Table 8.7.4.5 – MapHorzCoordSystemEventExtension Event Structures: A661_EVT_MAPHORZ_PROJECTION_REFERENCE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPHORZ_PROJECTION_REFERENCE
UnusedPad	N/A	16	0
Projection Reference Point Latitude	fr(180)	32	
Projection Reference Point Longitude	fr(180)	32	

8.7.5 MapVertCoordSystemEventExtension

Description:
The MapVertCoordSystemEventExtension is an extension which allows the CDS to report changes to the range and Projection Reference Point (RefGeoPosX and RefGeoPosY) on a MapVert widget (for example a pinch interaction could trigger a A661_EVT_VERT_RANGE_X or A661_EVT_VERT_RANGE_Y event).

The UA can choose (via the EventIDArray parameter) which events it wants to receive among the available events added by this extension.

The availability of these events for the CDS, and the way the CDS interprets the user interaction to send the values, are implementation dependent. For example,the event could be sent continuously when the user performs a drag and drop on the map, or only at the end of the gesture.

Note that this extension is available on both the MapVert, MapVert_Source and MapExternalSource widgets. When this extension is applied on a MapVert_Source or MapExternalSource widget, it sends the events corresponding to user interactions on the MapVert parent of the MapVert_Source or MapExternalSource.

MapVertCoordSystemEventExtension Parameters are defined in Table 8.7.5-1.

Table 8.7.5-1 – MapVertCoordSystemEventExtension Parameters

Parameters	Change	Description
ExtensionType	D	A661_MAPVERT_COORD_SYSTEM_EVENT_EXTENSION
NumEvents	D	The number of events in the following EventsArray
EventIdArray	D	Array to define which events the UA wants to receive using the enumerations below. A661_RANGE A661_PROJECTION_X_REFERENCE A661_PROJECTION_Y_REFERENCE
ReportMode	D	A661_REPORT_ON_CDS Report only changes originating from the CDS A661_REPORT_ALL Report all changes

MapVertCoordSystemEventExtension Creation Structure is defined in Table 8.7.5-2.

Table 8.7.5-2 – MapVertCoordSystemEventExtension Creation Structure

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
ExtensionType	ushort	16	A661_MAPVERT_COORD_SYSTEM_EVENT_EXTENSION
NumEvents	ushort	16	
ReportMode	uchar	8	A661_REPORT_ON_CDS A661_REPORT_ALL
UnusedPad	N/A	24	0
EventIdArray[NumEvents]	{ushort}*	16 * NumEvents + Pad	There are NumEvents events in the array. The complete array is followed by 2 bytes padding if NumEvents is an odd number.

A661_EVT_VERT_RANGE_X is defined in Table 8.7.5.3

Table 8.7.4.3 – MapVertCoordSystemEventExtension Event Structures: A661_EVT_MAPVERT_RANGE_X

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPVERT_RANGE_X
UnusedPad	N/A	16	0
RangeX	fr(32768)	32	X range expressed in nm

A661_EVT_VERT_RANGE_Y is defined in Table 8.7.5.4

Table 8.7.4.4 – MapVertCoordSystemEventExtension Event Structures: A661_EVT_MAPVERT_RANGE_Y

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPVERT_RANGE_Y
UnusedPad	N/A	16	0
RangeY	long	32	Y range expressed in feet

A661_EVT_VERT_PROJECTION_REFERENCE is defined in Table 8.7.4.5

Table 8.7.4.5 – MapVertCoordSystemEventExtension Event Structures: A661_EVT_MAPVERT_PROJECTION_REFERENCE

EventStructure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	A661_EVT_MAPVERT_PROJECTION_REFERENCE
UnusedPad	N/A	16	0
RefGeoPosX	fr(32768)	32	Geo-referenced Position X of the reference point, expressed in nm
RefGeoPosY	long	32	Geo-referenced Position Y of the reference point, expressed in feet

9.0 ANIMATION LAWS DEFINITION

9.1 Introduction

This section details how animation laws can be defined in Definition Files. These laws are used in the AnimationTranslation, AnimationRotation, AnimationScale, and AnimationOnParam widgets, and are identified by an AnimationLawRef parameter on those widgets.

Similar to vector symbols defined by the Symbol Graphical Definition language (see Section 5.0 of this specification), animation laws defined in a Definition File can be referenced by any layer whose User Application Layer Definition (UALD) is defined in the same Definition File as the animation law it wants to reference.

The following scheme should be used to find the appropriate animation law for a given widget AnimationLawRef parameter value:

- If an animation law with matching ID is found in the DF, use it
- Otherwise, if there is a global animation law with that ID defined in the CDS, use it
- Otherwise, behavior is implementation-dependent: perhaps a default easing function such as “Linear” would be used

9.2 Animation Law Definition Structures

Animation laws are defined in an AnimationLawBlock structure, defined in Table 9.2-1.

Table 9.2-1 – Animation Law Block Structure Exchanged Between UA and CDS at Definition Time

A661_Animation_Law_Block_Structure_DT	Type	Size (bits)	Description
A661_BEGIN_ANIMATION_LAW_BLOCK	uchar	8	Start keyword opening an animation law block
UnusedPad	N/A	24	0
Block Size	ulong	32	Size of this block, including header, in bytes
{A661_Create_Animation_Law_Structure}+	N/A	{32}+	Set of command structures, as applicable
A661_END_ANIMATION_LAW_BLOCK	uchar	8	Keyword ending an animation law block
UnusedPad	N/A	24	0

The creation structures found within the Animation Law Block are defined in Table 9.2-2.

Table 9.2-2 – Animation Law CommandStructure Exchanged Between UA and CDS at Definition Time

A661_Create_Animation_Law_Structure	Type	Size (bits)	Description
A661_CMD_CREATE_ANIMATION_LAW	ushort	16	Start keyword for opening the create structure
CommandSize	ushort	16	Size of the command, in bytes
AnimationLawId	ushort	16	Animation Law ID Value
UnusedPad	N/A	16	0
CreateParameterBuffer	N/A	{32}+	Refer to Section 9.3 for the description of the creation parameter buffers

9.3 Animation Law Commands

This section describes animation law definition commands used to define an animation law.

In this supplement, there is only one animation law command: A661_ANIMATION_LAW_DEFN_EASING_CURVE. That could change in future Supplements to this standard.

9.3.1 Easing Curve Animation Law

An easing curve animation law defines the animation law in terms of a ControlPoints array which defines an easing curve composed of one or more cubic Bezier curves attached together. The standard definition of a single cubic Bezier curve is defined by four control points (passing through P0 and P3):

- P0: the start point
- P1, P2: control points adjusting curvature
- P3: the end point

To avoid redundancy, the ControlPoints array does ***not*** include:

- Any P0 entries: The P0 of the first Bezier curve is implicitly (0,0) and the P0 of any other Bezier curve is implicitly the same as P3 of the preceding Bezier.
- P3 of the last cubic Bezier curve, which is implicitly (1,1).

The following restrictions limit the X value range and ensure that for any legal value of X there is a single Y value defined by the easing curve. The ControlPoints must be consistent with these restrictions for the easing curve animation law to be valid. Handling of invalid easing curve animation laws is implementation-dependent.

- The X coordinates of the control points lie in the interval [0 ..1].
- P0.X < P3.X < 1.
- P0.X ≤ P1.X ≤ P3.X.
- P0.X ≤ P2.X ≤ P3.X.

There is no specific restriction on control point Y coordinate values. They typically have values covering the entire [0 .. 1] range. They can also go outside that range to achieve an “overshoot” effect. See Section 9.5 for examples.

Table 9.3.1-1 – A661_ANIMATION_LAW_DEFN_EASING_CURVE Command Structure

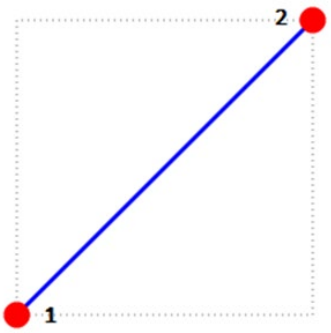
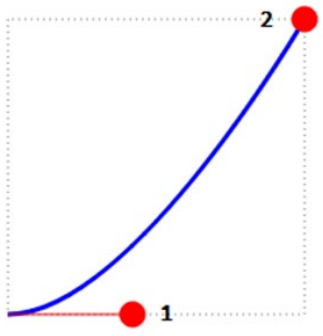
Field Name	Type	Size (bits)	Description
AnimationLawCommandType	ushort	16	A661_ANIMATION_LAW_DEFN_EASING_CURVE
UnusedPad	ushort	16	0
ControlPoints	{float, float}+	{32}+	Contains the control points, excluding the first at (0,0) and last at (1,1).

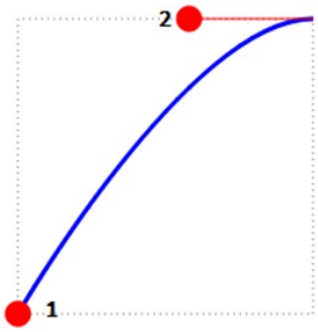
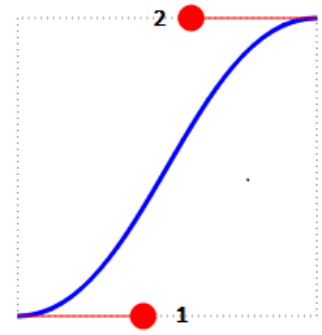
9.4 Pre-Defined CDS Animation Laws

Animation Law tables are different from other tables (e.g., Picture or Symbol tables) in that there are some ID values that are reserved to have special meaning. Table 4.6-14 describes two ranges of Animation Law ID values:

- There is a range of implementation dependent IDs available. In other tables (e.g., Picture, Symbol tables) all IDs are implementation dependent.
- Predefined IDs are found in the “Specification Range” and correspond to standard cases of animation laws. The specific definition of the animation laws with these IDs is left implementation-dependent. These ID values can be defined globally in an implementation-dependent way and/or in Animation Law tables defined within DFs. Definitions of those predefined IDs within DFs could allow different definitions of those laws to be specified in different contexts within the same CDS.
  Table 9.4-1 describes the predefined ID values. In the illustrative images, circles represent P1 and P2 control points, numbered according to the order in which they occur in ControlPoints.

Table 9.4-1 – Pre-Defined CDS Animation Laws

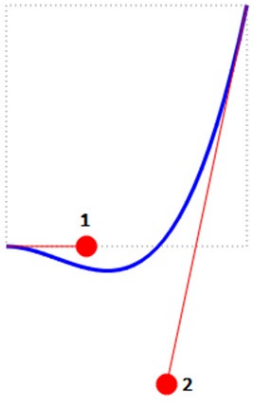
Predefined ID Value	Description	Illustrative Image
A661_LINEAR	Specifies an animation with a constant speed based on duration. It could be defined as an easing curve animation with ControlPoints: 1: (0,0) 2: (1,1)	
A661_EASING_IN	Specifies an easing curve animation with a slower start.	

Predefined ID Value	Description	Illustrative Image
A661_EASING_OUT	Specifies an easing curve animation with a slower end.	
A661_EASING_INOUT	Specifies an easing curve animation with a slower start and a slower end.	

9.5 Examples

The following examples are provided to help illustrate easing curve usage for slightly more interesting curves than the standard ones shown above. They are only provided for illustrative purposes (they are not standard ARINC 661 easing curve definitions). In all cases, the bottom-left corner and top right corner of the largest rectangle have coordinates (0,0) and (1,1), respectively. As explained above, those coordinates are implicit and not found in the easing curve definition. The control points are found in the DF file according to their order along the curve. The circular control points represent P1, P2: control points adjusting curvature. The square control points represent P3 control points. The control points are numbered according to the order in which they occur in ControlPoints.

The following easing curve includes a control point that has a negative y value:



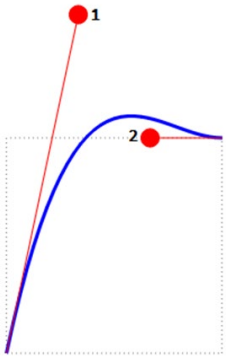
ControlPoints:

1: (0.3333,0)

2: (0.6666,-0.5725)

Figure 9.5-1 – Easing Curve

The following easing curve includes a control point that has a y value greater than one:



ControlPoints:

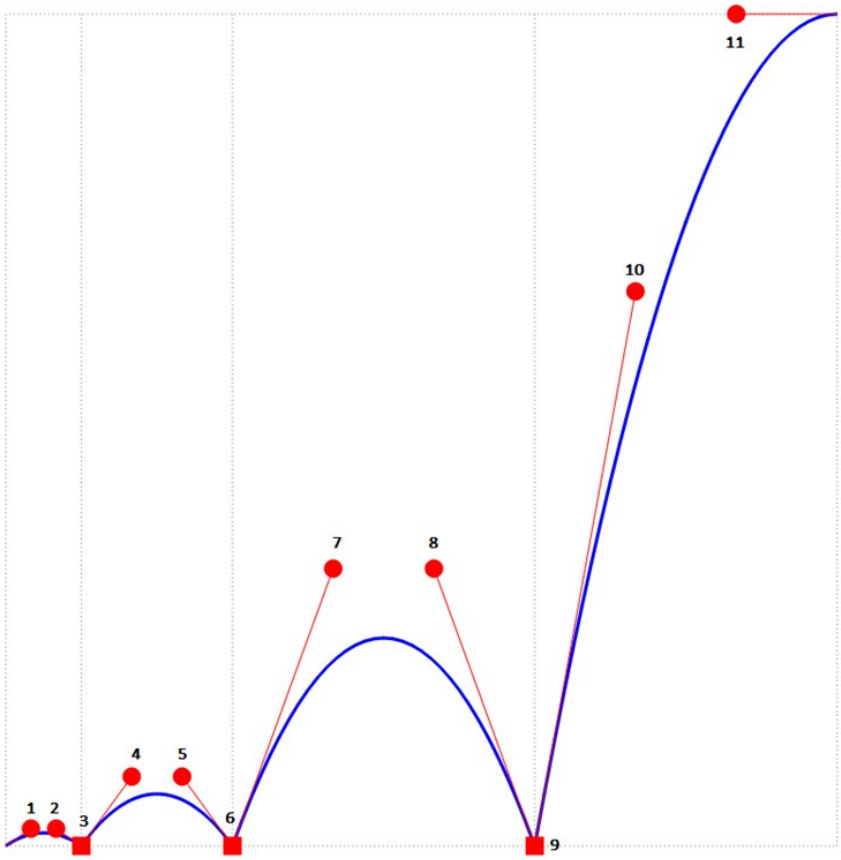
1: (0.3333,1.5725)

2: (0.6666, 1)

Figure 9.5-2 – Easing Curve

The following easing curve consists of 4 cubic Bezier curves attached together at (square) P3 control points. Vertical lines highlight those divisions.

Note: P3 control points do not necessarily have y=0.



ControlPoints:

1: (0.0303, 0.0208)
2: (0.0606, 0.0208)
3: (0.0909, 0)
4: (0.1515, 0.0833)
5: (0.2121, 0.0833)
6: (0.2727, 0)
7: (0.3939, 0.3333)
8: (0.5151, 0.3333)
9: (0.6363, 0)
10: (0.7575, 0.6666)
11: (0.8787, 1)

Figure 9.5-3 – Easing Curve

The following animationlawtable definition (as would be found within an XML DF) defines the three animation laws described immediately above with implementation dependent ids of 0x8000, 0x8001, and 0x8002, respectively.

```
<animationlawtable>

  <animationlawdefncmd name="EaseInBack"
type=”A661_ANIMATION_LAW_DEFN_EASING_CURVE”>

    <model>
```

```
        <prop name="AnimationLawId" value="0x8000"/>
        <arrayprop name="ControlPoints">
          <xyentry x="0.3333" y="0"/>
          <xyentry x="0.6666" y="-0.5725"/>
        </arrayprop>
      </model>
    </animationlawdefncmd>
    <animationlawdefncmd name="EaseOutBack"
  type="A661_ANIMATION_LAW_DEFN_EASING_CURVE">
      <model>
        <prop name="AnimationLawId" value="0x8001"/>
        <arrayprop name="ControlPoints">
          <xyentry x="0.3333" y="1.5725"/>
          <xyentry x="0.6666" y="1"/>
        </arrayprop>
      </model>
    </animationlawdefncmd>
    <animationlawdefncmd name="EaseInBounce"
  type="A661_ANIMATION_LAW_DEFN_EASING_CURVE">
      <model>
        <prop name="AnimationLawId" value="0x8002"/>
        <arrayprop name="ControlPoints">
          <xyentry x="0.0303" y="0.0208"/>
          <xyentry x="0.0606" y="0.0208"/>
          <xyentry x="0.0909" y="0"/>
          <xyentry x="0.1515" y="0.0833"/>
          <xyentry x="0.2121" y="0.0833"/>
          <xyentry x="0.2727" y="0"/>
          <xyentry x="0.3939" y="0.3333"/>
          <xyentry x="0.5151" y="0.3333"/>
          <xyentry x="0.6363" y="0"/>
          <xyentry x="0.7575" y="0.6666"/>
          <xyentry x="0.8787" y="1"/>
        </arrayprop>
      </model>
    </animationlawdefncmd>
</animationlawtable>
```

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 manages the cockpit displays. CDS performs many of the functions described in this document.

Crew member interaction (or crew member input)

Action of a crew member on an interactive widget through the use of an input device.

Cursor

Visual indicator of the point of crew input on the screen.

Definition File (DF)

A Definition File is associated with one UA. The DF contains the UA Layer Description (UALD) to be displayed on the CDS.

Definition phase

Consists in loading of DF in the CDS, which specify the creation of widgets in order to describe user application's interface layouts. The instantiation (creation plus first setting of all parameters) of widgets inside the CDS is part of the definition phase. This will occur before the beginning of the run-time phase.

Disabled

State of an interactive widget when it does not react to crew-member activation.

Display

A non-specific term, generally meaning "thing you look at." As a noun, "display" refers to a physical assembly of metal and glass (Display Head), or to the pattern of colored dots that appear on that glass (Format). As a verb, "display" refers to the act of deciding which of those colored dots to light (render) or, loosely, to providing the parameters necessary to be able to render.

Display Head

A physical assembly that can generate patterns of colored dots (raster) or lines (stroke).

Event

Notification sent by the CDS to a UA to indicate a crew member interaction has occurred on a widget owned by that UA.

APPENDIX A
GLOSSARY

Focus

State of a widget in which this widget receives the inputs 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 a highlighted widget will bring the focus on this widget (refer to Focus). Depending on the OEM choice, the click may also select the widget.

Identifier

A numeric value or tuple representing an ARINC 661 object (such as a widget or layer). See Section 3.1.1.

Inner state

Specific states of a widget. This state level represents the core of the widget behavior as well as its functional objectives. Examples of inner states:

For a basic PushButton, there is one inner state.

For a CheckButton, there are two inner states, which are ‘SELECTED’ and ‘UNSELECTED’.

Interactive

Category of widgets that can generate events in response to crew member activity on input device(s).

Layer

Layers provide the mechanism to combine graphical information from several UAs inside one window. For example, layers of the ND image include compass rose, map with interactive way-points, TCAS information, terrain or weather display. A layer is connected to a unique UA, whereas a UA can use several layers.

A layer is the higher level entity of the CDS that is known by the UA. From the UA point of view, the Layer is the high level container in the hierarchical structure of the UA widgets. From the CDS point of view, the layer is one graphical layer associated with one UA inside a window. The layer layout within a window is beyond the scope of this standard. Refer to Section 2.3.2, Layer Definition.

Look and 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.

APPENDIX A
GLOSSARY

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

An Identifier, or a unique CDS-wide index number that resolves to an Identifier. A Reference refers to 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 and 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)

Equipment that manages a function and owns layers. UAs communicate with the CDS in order to have their layers rendered on the cockpit displays.

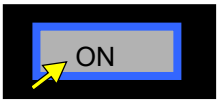
APPENDIX A
GLOSSARY

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 refer to “Look and Feel,” and a behavior.

Window

A window defines a rectangular physical area of the display surface. A window consists of one or more layers and is controlled by the CDS.

APPENDIX B ACRONYMS AND ABBREVIATIONS

ADI	Attitude Director Indicator
AEEC	Airlines Electronic Engineering Committee
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
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
DTD	Document Type Declaration
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
GBK	Guojia Biaozhun Kuozhan (Simplified Chinese character set)
GG	Graphic Generator
GNLU	GNSS Navigation and Landing Unit
GNU	GNSS Navigation Unit
Gp	Graphic primitive
HMI	Human Machine Interface
HSI	Horizontal Situation Indicator
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

APPENDIX B
ACRONYMS AND ABBREVIATIONS

UADF	User Application Definition File
UALD	User Application Layer Definition
UTF	Unicode Transformation Format
WXR	Weather Radar
XML	Extensible Markup Language
XSD	XML Schema Document
2D	Two Dimensional
3D	Three Dimensional

APPENDIX C EXAMPLE OF A DEFINITION FILE

C-1 User Application Example Overview

The following UA example controls the cabin temperature using two interfaces:

1. The UA is connected to the aircraft environment with:
   a. One input: a cabin temperature sensor
   b. One output: an actuator for the heater system and cooler system
2. The UA is connected to the CDS with an ARINC 661 interface to display the following format in a DU window:

Figure C-1.1 – Cabin Temperature DU Window

Buttons increment or decrement a selected temperature for the cabin, indicated by the small green pointer. The current cabin temperature is indicated by both the white arrow and the digital readout.

The format is composed of eight ARINC 661 widgets summarized as follows:

The scale is an ARINC 661 picture (A661_PICTURE : several colors, no rotation).

Pointers (selected and real temperature) are ARINC 661 symbols (A661_SYMBOL: can rotate, has one selected color).

The digital readout and its unit string are ARINC 661 labels (A661_LABEL).

Buttons are ARINC 661 buttons using a reference to a symbol (A661_PICTURE_PUSH_BUTTON).

All drawings are clipped inside an ARINC 661 panel container (A661_PANEL).

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

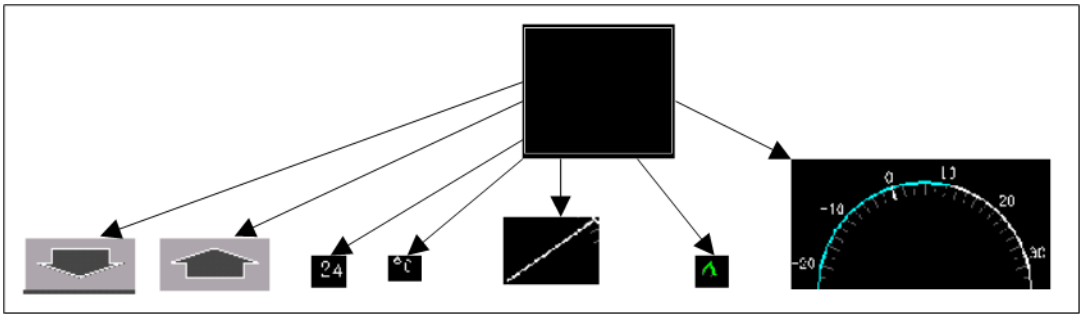


Figure C-1.2 – Cabin Temperature Hierarchical Structure

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-2 Example of Application Code at Run Time

An example of UA code for controlling the cabin temperature example at run time follows:

```
if ((selectedCabinTemp < maxValue) and (BUTTON_PRESSED.id = IncreaseSelectTemp)) then
       increment(selectedCabinTemp);

end if
if ((selectedCabinTemp > minValue) and (BUTTON_PRESSED.id = DecreaseSelectTemp)) then
       decrease(selectedCabinTemp);

end if
Actuator.heaterCommand = PIDcontroller (selectedCabinTemp);
angleValue = selectedCabinTemp * scaleFactor + offset;
setParameter(TemperatureSelectedPointer, RotationAngle, angleValue);
       cockpitTemp = Sensor.cabinTemp
if IsValid(cockpitTemp) = True then
       angleValue = cockpitTemp * scaleFactor + offset;

       setParameter(IndicatedTempDRO, LabelString, toString(cockpitTemp));
       setParameter(TemperatureIndicatedPointer, RotationAngle, angleValue);
       if (cockpitTemp > ThresholdValue ) then
              setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_WARNING);
              setParameter(TemperatureIndicatedPointer, StyleSet,
A661_STYLE_SET_WARNING);
else
              setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_NOMINAL);
              setParameter(TemperatureIndicatedPointer, StyleSet,
A661_STYLE_SET_NOMINAL);

       end if
setParameter(TemperatureIndicatedPointer, Visible, A661_TRUE);
       setParameter(IncreaseSelectTemp, Enable, A661_TRUE);
       setParameter(DecreaseSelectTemp, Enable, A661_TRUE);
else
       setParameter(IndicatedTempDRO, LabelString, "---");
       setParameter(IndicatedTempDRO, StyleSet, A661_STYLE_SET_WARNING);
       setParameter(TemperatureIndicatedPointer, Visible, A661_FALSE);
       beginBlock();
       setParameter(IncreaseSelectTemp, Enable, A661_FALSE);
       setParameter(DecreaseSelectTemp, Enable, A661_FALSE);
       endBlock();
end if
```

The above is only an example of UA messaging logic. See Appendix I for UA design considerations.

Begin-Block and End-Block commands limit the amount of data that can be processed as coherent information. In the following example, the corresponding byte stream is sent for this block by the UA assuming that the network allows transmission of blocks of such size. In other cases, sub-blocks may 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
4566                    # WIDGET ID (IncreaseSelectTemp)
0000                    # UNUSED PAD

                        # 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
4567                    # WIDGET ID (DecreaseSelectTemp)
0000                    # UNUSED PAD

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

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

APPENDIX C
EXAMPLE OF A DEFINITION FILE

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_EVT_SELECTION)
0000                    # UNUSED PAD

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

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

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.

Note: Unit length of measure is 1/100 of millimeter.

UADF Example for Cabin Temperature Application

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

                              # Word 1
A661                          # A661_DF_MAGIC_NUMBER
00                            # A661 LIBRARY VERSION
02                            # A661 SUPP VERSION

                              # Word 2
6788                          # DF ID
0000                          # SIZE OF OEM FREE DATA

                              # Word 3
A0                            # A661_BEGIN_LAYER_BLOCK
42                            # LAYER ID
1230                          # CONTEXT NUMBER

                              # Word 4
0000013C                      # BLOCK SIZE (=316 bytes, words 3 - 81)

                              # Widget instance number:1
                              # Word 5
CA01                          # A661_CMD_CREATE
0020                          # COMMAND SIZE (= 32 bytes, words 5 - 12)
A1F0                          # WIDGET TYPE (A661_PANEL)
1221                          # WIDGET ID (CabinTempPanel)
0000                          # PARENT ID (zero indicates layer is parent)
01                            # Enable, value:A661_TRUE
01                            # Visible, value:A661_TRUE
00002AF8                      # PosX, value: 11000 = 110 mm, 4.33 in
0000319C                      # PosY, value: 12700 = 127 mm, 5 in
00001DC4                      # SizeX, value: 7620 = 76.2 mm, 3 in
0000175A                      # SizeY, value: 5978 = 59.78 mm, 2.44 in
0000                          # StyleSet, value:A661_STYLE_SET_DEFAULT
00                            # MotionAllowed, value: A661_FALSE
00                            # UNUSED PAD

                              # Widget instance number:2
                              # Word 13
CA01                          # A661_CMD_CREATE
0020                          # COMMAND SIZE (=32 bytes, words 13 - 20)
A200                          # WIDGET TYPE (A661_PICTURE)
1222                          # WIDGET ID (TemperatureCelciusScale)
1221                          # PARENT ID (CabinTempPanel)
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
00                          # Anonymous, value:A661_FALSE
01                          # Visible, value:A661_TRUE
000002EE                    # PosX, value: 750, 7.5 mm, 0.31 in
00000956                    # PosY, value: 2390, 23.9 mm, 0.98 in
000017EE                    # SizeX, value: 6126, 61.26 mm, 2.41 in
000009BA                    # SizeY, value: 2490, 24.9 mm, 1.02 in
0000                        # StyleSet, value: A661_STYLE_SET_DEFAULT
9870                        # PictureRef

                            # Widget instance number:3
                            # Word 21
CA01                        # A661_CMD_CREATE
0020                        # COMMAND SIZE (=32 bytes, words 21 - 28)
A310                        # WIDGET TYPE (A661_SYMBOL)
1223                        # WIDGET ID (TemperatureIndicatedPointer)
1221                        # PARENT ID (CabinTempPanel)
00                          # MotionAllowed, value: A661_TRUE
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
238E38E3                    # RotationAngle, value: 596523235 = 50 deg
                            [fr(180)]
0001                        # StyleSet, value: OEM_STYLESET_FREE_COLOR
9874                        # SymbolReference,
                            value:SymbolTemperatureIndicatedPointer
0F                          # ColorIndex, value: OEM_WHITE
000000                      # UNUSED PAD

                            # Widget instance number:4
                            # Word 29
CA01                        # A661_CMD_CREATE
0020                        # COMMAND SIZE (=32 bytes, words 29 - 36)
A310                        # WIDGET TYPE (A661_SYMBOL)
1224                        # WIDGET ID (TemperatureSelectedPointer)
1221                        # PARENT ID (CabinTempPanel)
00                          # MotionAllowed, value: A661_TRUE
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
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
071C71C7                    # RotationAngle, value: 119304647 = 10 deg
                            [fr(180)]
0001                        # StyleSet, value: OEM_STYLESET_FREE_COLOR
0003                        # SymbolReference,
                            value:SymbolTemperatureSelectedPointer
01                          # ColorIndex, value: OEM_GREEN4
000000                      # UNUSED PAD

                           # Widget instance number:5
                           # Word 37
CA01                       # A661_CMD_CREATE
2C                         # COMMAND SIZE (=44 bytes, words 37 - 47)
A160                       # WIDGET TYPE (A661_LABEL)
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)]
0801                       # StyleSet, value:OEM_STYLESET_NORMAL_READOUT
0004                       # MaxStringLength, value:4
00                         # MotionAllowed, value:A661_FALSE
00                         # Font, value: OEM_STYLESET_DEFAULT_FONT
00                         # UNUSED PAD
14                         # Alignment, value:A661_RIGHT
32340000                   # LabelString, value: "24"

                           # Widget instance number:6
                           # Word 48
CA01                       # A661_CMD_CREATE
2C                         # COMMAND SIZE (=44 bytes, words 48 - 58)
A160                       # WIDGET TYPE (A661_LABEL)
1226                       # WIDGET ID (IndicatedUnitLabel)
1221                       # PARENT ID (CabinTempPanel)
00                         # Anonymous, value:A661_FALSE
01                         # Visible, value:A661_TRUE
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
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)]
0801                        # StyleSet, value:OEM_STYLESET_NORMAL_READOUT
0002                        # MaxStringLength, value:2
00                          # MotionAllowed, value:A661_FALSE
00                          # Font, value: OEM_STYLESET_DEFAULT_FONT
00                          # UNUSED PAD
13                          # Alignment, value:A661_LEFT
81430000                    # LabelString, value: "°C"

                            # Widget instance number:7
                            # Word 59
      CA01                  # A661_CMD_CREATE
      0020                  # COMMAND SIZE (=32 bytes, words 62 - 69)
      A240                  # WIDGET TYPE (A661_PICTURE_PUSH_BUTTON)
      1227                  # WIDGET ID (IncreaseSelectTemp)
      1221                  # PARENT ID (CabinTempPanel)
      01                    # Enable, value:A661_TRUE
      01                    # Visible, value:A661_TRUE
      000001D9              # PosX, value: 473, 4.73 mm, 0.19 in
      000001D9              # PosY, value: 473, 4.73 mm, 0.19 in
      00000B30              # SizeX, value: 2864, 28.64 mm, 1.13 in
      000003B6              # SizeY, value: 950, 9.5 mm, 0.39 in
      0000                  # StyleSet, value:A661_STYLE_SET_DEFAULT
      0000                  # NextFocusedWidget, value:0
      9878                  # PictureReference, value:SymbolArrowUp
      0000                  # MaxStringLength, value:0
      15                    # PicturePosition, value:A661_CENTER
      00                    # AutomaticFocusMotion, value:A661_FALSE
      00                    # Alignment, value: A661_CENTER
      00                    # UNUSED PAD
      00000000              # LabelString, value: ""

                            # Widget instance number:8
                            # Word 70
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

UADF Example for Cabin Temperature Application

```
A01                     # A661_CMD_CREATE
002C                    # COMMAND SIZE (=44 bytes, words 70 - 80)
A240                    # WIDGET TYPE (A661_PICTURE_PUSH_BUTTON)
1228                    # WIDGET ID (DecreaseSelectTemp)
1221                    # PARENT ID (CabinTempPanel)
01                      # Enable, value:A661_TRUE
01                      # Visible, value:A661_TRUE
00000B30                # PosX, value: 3984, 39.84 mm, 1.63 in
000001D9                # PosY, value: 473, 4.73 mm, 0.19 in
00000B30                # SizeX, value: 2864, 28.64 mm, 1.13 in
000003B6                # SizeY, value: 950, 9.5 mm, 0.39 in
0000                    # StyleSet, value:A661_STYLE_SET_DEFAULT
0000                    # NextFocusedWidget, value:0
987C                    # PictureReference, value:SymbolArrowDown
0000                    # MaxStringLength, value:0
15                      # PicturePosition, value:A661_CENTER
00                      # AutomaticFocusMotion, value:A661_FALSE
00                      # Alignment, value: A661_CENTER
00                      # UNUSED PAD
00000000                # LabelString, value: “”
                        # Word 81
C0                      # A661_END_LAYER_BLOCK
000000                  # UNUSED PAD
E0                      # A661_DF_FOOTER
000000                  # UNUSED PAD
# END DF
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

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

For the application illustrated in Section C-3, the corresponding XML form of the DF is as follows. Refer to Section 6.0 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values.

```
<?xml version="1.0" encoding="UTF-8"?>

<a661_df library_version="0" supp_version="6" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://www.aviation-ia.com/support-files/661-7-df/1"
xsi:schemaLocation="http://www.aviation-ia.com/support-files/661-7-df/1 a661.xsd">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <a661_layer>
      <model>
         <prop name="LayerId" value="66"/>
         <prop name="ContextNumber" value="0x1230"/>
         <prop name="Height" value="20000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="CabinTempPanel" type="A661_PANEL">
         <model>
            <prop name="WidgetIdent" value="4641"/>
            <prop name="PosX" value="11000"/>
            <prop name="PosY" value="12700"/>
            <prop name="SizeX" value="7620"/>
            <prop name="SizeY" value="5978"/>
         </model>
         <a661_widget name="TemperatureCelciusScale" type="A661_PICTURE">
            <model>
               <prop name="WidgetIdent" value="4642"/>
               <prop name="PosX" value="750"/>
               <prop name="PosY" value="2390"/>
               <prop name="SizeX" value="6126"/>
               <prop name="SizeY" value="2490"/>
               <prop name="PictureReference" value="SymbolTemperatureCelciusScale"/>
            </model>
         </a661_widget>
         <a661_widget name="TemperatureIndicatedPointer" type="A661_SYMBOL">
            <model>
               <prop name="WidgetIdent" value="4643"/>
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="596523235" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_WHITE" />
               <prop name="SymbolReference" value="SymbolTemperatureIndicatedPointer" />
            </model>
         </a661_widget>
         <a661_widget name="TemperatureSelectedPointer" type="A661_SYMBOL">
            <model>
               <prop name="WidgetIdent" value="4644"/>
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="119304647" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_GREEN4" />
               <prop name="SymbolReference" value="SymbolTemperatureSelectedPointer" />
            </model>
         </a661_widget>
         <a661_widget name="IndicatedTempDRO" type="A661_LABEL">
            <model>
               <prop name="WidgetIdent" value="4645"/>
               <prop name="PosX" value="2864" />
               <prop name="PosY" value="1768" />
               <prop name="SizeX" value="950" />
               <prop name="SizeY" value="622" />
               <prop name="StyleSet" value="OEM_STYLESET_NORMAL_READOUT" />
               <prop name="Font" value="OEM_STYLESET_DEFAULT_FONT" />
               <prop name="MaxStringLength" value="4" />
               <prop name="Alignment" value="A661_RIGHT" />
               <prop name="LabelString" value="24" />
            </model>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
            </a661_widget>
            <a661_widget name="IndicatedUnitLabel" type="A661_LABEL">
               <model>
                  <prop name="WidgetIdent" value="4645"/>
                  <prop name="PosX" value="3984" />
                  <prop name="PosY" value="1768" />
                  <prop name="SizeX" value="473" />
                  <prop name="SizeY" value="622" />
                  <prop name="StyleSet" value="OEM_STYLESET_NORMAL_READOUT" />
                  <prop name="Font" value="OEM_STYLESET_DEFAULT_FONT" />
                  <prop name="MaxStringLength" value="3" />
                  <prop name="LabelString" value="°C" />
               </model>
            </a661_widget>
            <a661_widget name="IncreaseSelectTemp" type="A661_PICTURE_PUSH_BUTTON">
               <model>
                  <prop name="WidgetIdent" value="4646"/>
                  <prop name="PosX" value="473" />
                  <prop name="PosY" value="473" />
                  <prop name="SizeX" value="2864" />
                  <prop name="SizeY" value="950" />
                  <prop name="PictureReference" value="SymbolArrowUp" />
               </model>
            </a661_widget>
            <a661_widget name="DecreaseSelectTemp" type="A661_PICTURE_PUSH_BUTTON">
               <model>
                  <prop name="WidgetIdent" value="4647"/>
                  <prop name="PosX" value="3984" />
                  <prop name="PosY" value="473" />
                  <prop name="SizeX" value="2864" />
                  <prop name="SizeY" value="950" />
                  <prop name="PictureReference" value="SymbolArrowDown" />
               </model>
            </a661_widget>
         </a661_widget>
      </a661_layer>
   </a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-5 A More Interesting XML DF Example

This is an XML file example that uses a wider range of property types. Refer to Section 6.0 for a description of the XML DF format, including the ways that DefaultStyleText can be represented. In this example not all property values are given explicitly. The unspecified properties take default values.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="3" supp_version="2">
   <model>
      <prop name="ApplicationId" value="45"/>
   </model>
   <a661_layer name="ATC_COM">
      <model>
         <prop name="LayerId" value="55"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="10000"/>
         <prop name="Width" value="10000"/>
      </model>
      <a661_widget name="PNL_COM" type="A661_PANEL">
         <model>
            <prop name="WidgetIdent" value="1"/>
            <prop name="Visible" value="A661_TRUE"/>
            <prop name="Enable" value="A661_TRUE"/>
            <prop name="StyleSet" value="0"/>
            <prop name="PosX" value="100"/>
            <prop name="PosY" value="100"/>
            <prop name="SizeX" value="7000"/>
            <prop name="SizeY" value="5000"/>
         </model>
         <a661_widget name="CKB_CHOICE" type="A661_CHECK_BUTTON">
            <model>
               <prop name="WidgetIdent" value="2"/>
               <prop name="Visible" value="A661_TRUE"/>
               <prop name="Enable" value="A661_TRUE"/>
               <prop name="CheckButtonState" value="A661_UNSELECTED"/>
               <prop name="StyleSet" value="1"/>
               <prop name="PosX" value="100"/>
               <prop name="PosY" value="100"/>
               <prop name="SizeX" value="3000"/>
               <prop name="SizeY" value="900"/>
               <prop name="NextFocusedWidget" value="0"/>
               <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
               <prop name="LabelSring" value="SELECT"/>
               <prop name="MaxStringLength" value="7"/>
               <prop name="Alignment" value="A661_LEFT"/>
               <prop name="PicturePosition" value="A661_LEFT"/>
            </model>
         </a661_widget>
         <a661_widget name="SL_COUNTRY" type="A661_SCROLL_LIST">
            <model>
               <prop name="WidgetIdent" value="3"/>
               <prop name="Visible" value="A661_TRUE"/>
               <prop name="Enable" value="A661_FALSE"/>
               <prop name="StyleSet" value="2"/>
               <prop name="PosX" value="100"/>
               <prop name="PosY" value="1100"/>
               <prop name="SizeX" value="6800"/>
               <prop name="SizeY" value="3800"/>
               <prop name="NextFocusedWidget" value="0"/>
               <prop name="AutomaticFocusMotion" value="A661_FALSE"/>
               <prop name="NumberOfEntries" value="6"/>
               <prop name="MaxNumberOfEntries" value="15"/>
               <prop name="FirstVisibleEntry" value="3"/>
               <prop name="FirstAccessibleEntry" value="3"/>
               <prop name="FlagReportVisibleEntry" value="A661_FALSE"/>
               <prop name="SelectedEntry" value="3"/>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                <prop name="DefaultStyleText
value=”\x1B\x40\x00\x1B\x41\x00\x1B\x42\x29\x1B\x43\x00”/>
                <prop name="MaxStringLength" value="10"/>
                <prop name="Alignment" value="A661_CENTER"/>
                <arrayprop name="LabelStringArray">
                   <entry value="CANADA"/>
                   <entry value="USA"/>
                   <entry value="BELGIUM"/>
                   <entry value="FRANCE"/>
                   <entry value="GERMANY"/>
                   <entry value="UK"/>
                </arrayprop>
                <arrayprop name="EnableArray">
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                   <entry value="A661_TRUE"/>
                </arrayprop>
                <prop name="VerticalScroll" value="A661_RIGHT"/>
             </model>
          </a661_widget>
       </a661_widget>
       <a661_widget name="BF_COM" type="A661_BUFFER_FORMAT">
          <model>
             <prop name="WidgetIdent" value="4"/>
             <prop name="NumberOfFields" value="2"/>
             <arrayprop name="BufferStructure">
                <structentry>
                   <field name="WidgetIdent" value="2"/>
                   <field name="ParameterIdent" value="A661_INNER_STATE_CHECK"/>
                </structentry>
                <structentry>
                   <field name="WidgetIdent" value="3"/>
                   <field name="ParameterIdent" value="A661_ENABLE"/>
                </structentry>
             </arrayprop>
          </model>
       </a661_widget>
    </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-6 An XML DF Example With Extensions

This is an XML file example that uses extensions on widgets.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<a661_df name="extensionsExamplev3.xml" library_version="0" supp_version="2">
   <model>
      <prop name="ApplicationId" value="1" />
   </model>
   <a661_layer name="Default" >
      <model>
         <prop name="LayerId" value="1" />
         <prop name="ContextNumber" value="0" />
         <prop name="Height" value="10000" />
         <prop name="Width" value="10000" />
      </model>
      <a661_widget name="tabbedPanelGroup" type="A661_TABBED_PANEL_GROUP">
         <model>
            <prop name="WidgetIdent" value="1" />
            <prop name="Enable" value="A661_TRUE" />
            <prop name="Visible" value="A661_TRUE" />
            <prop name="PosX" value="0" />
            <prop name="PosY" value="0" />
            <prop name="SizeX" value="6000" />
            <prop name="SizeY" value="10000" />
            <prop name="StyleSet" value="0" />
            <prop name="ActiveTabbedPanelID" value="2" />
            <prop name="TabPosition" value="A661_TOP" />
            <prop name="AutomaticInsetSizeFlag" value="A661_TRUE" />
         </model>
         <a661_widget name="RegularTabbedPanel" type="A661_TABBED_PANEL">
            <model>
               <prop name="WidgetIdent" value="2" />
               <prop name="Enable" value="A661_TRUE" />
               <prop name="Visible" value="A661_TRUE" />
               <prop name="StyleSet" value="0" />
               <prop name="NextFocusedWidget" value="20" />
               <prop name="MaxStringLength" value="20" />
               <prop name="PictureReference" value="0" />
               <prop name="PicturePosition" value="A661_LEFT" />
               <prop name="AutomaticFocusMotion" value="A661_TRUE" />
               <prop name="Alignment" value="A661_CENTER" />
               <prop name="InsetSize" value="3000" />
               <prop name="LabelString" value="EXTENDED" />
            </model>
            <a661_widget name="EditBoxNumericWithExtension"
type="A661_EDIT_BOX_NUMERIC">
               <model>
                  <prop name="WidgetIdent" value="4" />
                  <prop name="Enable" value="A661_TRUE" />
                  <prop name="Visible" value="A661_TRUE" />
                  <prop name="StyleSet" value="0" />
                  <prop name="PosX" value="300" />
                  <prop name="PosY" value="-300" />
                  <prop name="SizeX" value="5000" />
                  <prop name="SizeY" value="1000" />
                  <prop name="NextFocusedWidget" value="0" />
                  <prop name="Value" value="0.0" />
                  <prop name="TicsCoarse" value="10.0" />
                  <prop name="TicsFine" value="1.0" />
                  <prop name="MinValue" value="0.0" />
                  <prop name="MaxValue" value="100.0" />
                  <prop name="LegendAreaSizeX" value="3000" />
                  <prop name="StartCursorPosByte" value="0" />
                  <prop name="MaxFormatStringLength" value="20" />
                  <prop name="MaxLegendStringLength" value="20" />
                  <prop name="LegendPosition" value="A661_LEFT" />
                  <prop name="AutomaticFocusMotion" value="A661_FALSE" />
                  <prop name="ReportAllChanges" value="A661_EDB_CHANGE_CONFIRMED" />
                  <prop name="Alignment" value="A661_CENTER" />
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                  <prop name="LegendRemoved" value="A661_TRUE" />
                  <prop name="NumericKeyFlag" value="A661_TRUE" />
                  <prop name="CyclicFlag" value="A661_TRUE" />
                  <prop name="FormatString" value="###" />
                  <prop name="LegendString" value="VAL" />
               </model>
               <a661_extension type="A661_DIRECTIONAL_TABBING_EXTENSION">
                  <model>
                     <prop name="NumDirections" value="2" />
                     <arrayprop name="TabDirectionWidgetIdArray">
                        <entry value="2" />
                        <entry value="5" />
                     </arrayprop>
                  </model>
               </a661_extension>
               <a661_extension type="A661_LEGEND_ALIGN_EXTENSION">
                  <model>
                     <prop name="LegendAlignment" value="A661_LEFT" />
                  </model>
               </a661_extension>
            </a661_widget>
         </a661_widget>
         <a661_widget name="ExtendedTabbedPanel" type="A661_TABBED_PANEL">
            <model>
               <prop name="WidgetIdent" value="3" />
               <prop name="Enable" value="A661_TRUE" />
               <prop name="Visible" value="A661_TRUE" />
               <prop name="StyleSet" value="0" />
               <prop name="NextFocusedWidget" value="20" />
               <prop name="MaxStringLength" value="20" />
               <prop name="PictureReference" value="0" />
               <prop name="PicturePosition" value="A661_LEFT" />
               <prop name="AutomaticFocusMotion" value="A661_TRUE" />
               <prop name="Alignment" value="A661_CENTER" />
               <prop name="InsetSize" value="3000" />
               <prop name="LabelString" value="DEFAULT" />
            </model>
            <a661_extension type="A661_DIRECTIONAL_TABBING_EXTENSION">
               <model>
                  <prop name="NumDirections" value="2" />
                  <arrayprop name="TabDirectionWidgetIdArray">
                     <entry value="3" />
                     <entry value="5" />
                  </arrayprop>
               </model>
            </a661_extension>
            <a661_widget name="RegularButton" type="A661_PUSH_BUTTON">
               <model>
                  <prop name="WidgetIdent" value="5" />
                  <prop name="Enable" value="A661_TRUE" />
                  <prop name="Visible" value="A661_TRUE" />
                  <prop name="PosX" value="300" />
                  <prop name="PosY" value="-300" />
                  <prop name="SizeX" value="2500" />
                  <prop name="SizeY" value="1000" />
                  <prop name="StyleSet" value="0" />
                  <prop name="NextFocusedWidget" value="0" />
                  <prop name="MaxStringLength" value="20" />
                  <prop name="AutomaticFocusMotion" value="A661_FALSE" />
                  <prop name="Alignment" value="A661_CENTER" />
                  <prop name="LabelString" value="DEFAULT" />
               </model>
            </a661_widget>
         </a661_widget>
      </a661_widget>
   </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-7 Binary DF Specifying Symbol Graphical Definitions

This section shows a binary Definition File that defines two symbols. See Section C-8 for an XML version of this example.

```
# Binary Definition File
# Example for ARINC-661 Specification
# Demonstrates how symbols and layers can be defined in
# a single binary definition file

A661            # A661_DF_MAGIC_NUMBER
02              # Library_Version
02              # Supp_Version
6788            # Application Identifier
0000            # UNUSED PAD

# Symbol Definition starts here:
F0              # A661_BEGIN_SYMBOL_BLOCK
000000          # UNUSED PAD
00000076        # block size (118 bytes)
CA04            # A661_CMD_CREATE_SYMBOL
0030            # command size (48 bytes)
0064            # symbol ID (100)
0000            # UNUSED PAD
90D0            # A661_SYMBOL_DEFN_POLYLINE
0004            # number of vertices
01              # closed (A661_TRUE)
000000          # UNUSED PAD
FFFFFE0C        # first X (-500)
FFFFFE0C        # first Y (-500)
000001F4        # second X (500)
000001F4        # second Y (500)
FFFFFE0C        # third X (-500)
000001F4        # third Y  (500)
000001F4        # fourth X (500)
FFFFFE0C        # fourth Y(-500)
CA04            # A661_CMD_CREATE_SYMBOL
003A            # command size (58 bytes)
0065            # symbol ID (101)
0000            # UNUSED PAD
90B0            # A661_SYMBOL_DEFN_LINE
0000            # UNUSED PAD
00000000        # first X (0)
FFFFFC18        # first Y (-1000)
00000000        # second X (0)
FFFFFE0C        # second Y (-500)
90F0            # A661_SYMBOL_DEFN_TRIANGLE
0000            # UNUSED PAD
00              # filled (A661_FALSE)
00              # UNUSED PAD
FFFFFF38        # first X (-200)
FFFFFE0C        # first Y (-500)
00000000        # second X (0)
00000000        # second Y (0)
000000C8        # third X (200)
FFFFFE0C        # third Y (-500)
F8              # A661_END_SYMBOL_BLOCK
000000          # UNUSED PAD
# Layer Definition starts here:
A0              # A661_BEGIN_LAYER_BLOCK
05              # layer ID (5)
0000            # context number
0000008C        # block size (140 bytes)
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A1F0            # widget type (A661_PANEL)
1221            # widget ID
0000            # parent ID (zero indicates layer is parent)
01              # enable, value: A661_TRUE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001DC4        # size_x, value: 7620
0000175A        # size_y, value: 5978
0000            # style_set
0000            # UNUSED PAD
CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1222            # widget ID
1221            # parent ID
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
000002EE        # pos_x, value: 750
00000956        # pos_y, value: 2390
000017EE        # size_x, value: 6126
000009BA        # size_y, value: 2490
0000            # style set
9870            # picture reference
CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A310            # widget type (A661_SYMBOL)
1223            # widget ID
1221            # parent ID
01              # motion allowed, value: A661_TRUE
01              # visible, value: A661_TRUE
00000EE2        # pos_x, value: 3810
00000956        # pos_y, value: 2390
238E38E3        # rotation angle, value: 596523235
0001            # style set, value: OEM_STYLESET_FREE_COLOR
0065            # symbol reference
0F              # color index, value: OEM_WHITE
000000          # UNUSED PAD

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A310            # widget type (A661_SYMBOL)
1224            # widget ID
1221            # parent ID
01              # motion allowed, value: A661_TRUE
01              # visible, value: A661_TRUE
00000D52        # pos_x, value: 3410
00000A82        # pos_y, value: 2690
071C71C7        # rotation angle, value: 119304647
0001            # style set, value: OEM_STYLESET_FREE_COLOR
0064            # symbol reference
01              # color index, value: OEM_GREEN
000000          # UNUSED PAD
C0              # A661_END_LAYER_BLOCK
000000          # UNUSED PAD
E0              # A661_DF_FOOTER
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-8 XML DF Specifying Symbol Graphical Definitions

This is the XML encoding of the example given in Section C-7. Refer to Section 6.0 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values. The WidgetId values are provided in hex format in this example. Both hex and decimal formats for unsigned integral values are allowed.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="2" supp_version="2">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <symboltable>
      <symboldefn name="HourGlassSymbol">
        <model>
           <prop name="Id" value="100" />
         </model>
         <stdrepr>
            <symboldefncmd type="A661_SYMBOL_DEFN_POLYLINE">
               <model>
                  <prop name="NumVertices" value="4" />
                  <prop name="Closed" value="A661_TRUE" />
                  <arrayprop name="Vertices">
                     <xyentry x="-500" y="-500" />
                     <xyentry x="500" y="500" />
                     <xyentry x="-500" y="500" />
                     <xyentry x="500" y="-500" />
                  </arrayprop>
               </model>
            </symboldefncmd>
         </stdrepr>
      </symboldefn>
      <symboldefn name="VertArrowSymbol">
         <model>
           <prop name="Id" value="101" />
         </model>
         <stdrepr>
            <symboldefncmd type="A661_SYMBOL_DEFN_LINE">
               <model>
                  <prop name="PosXStart" value= "0" />
                  <prop name="PosYStart" value="-1000" />
                  <prop name="PosXEnd" value="0" />
                  <prop name="PosYEnd" value="-500" />
               </model>
            </symboldefncmd>
            <symboldefncmd type="A661_SYMBOL_DEFN_TRIANGLE">
               <model>
                  <prop name="Filled" value="A661_FALSE" />
                  <prop name="PosX" value="-200" />
                  <prop name="PosY" value="-500" />
                  <prop name="PosX2" value="0" />
                  <prop name="PosY2" value="0" />
                  <prop name="PosX3" value="200" />
                  <prop name="PosY3" value="-500" />
               </model>

            </symboldefncmd>
         </stdrepr>
      </symboldefn>
   </symboltable>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
   <a661_layer name="SYMBOL_EXAMPLE_LAYER">
      <model>
         <prop name="LayerId" value="5"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="20000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="Panel" type="A661_PANEL">
         <model>
            <prop name="WidgetIdent" value="0x1221" />
            <prop name="PosX" value="11000" />
            <prop name="PosY" value="12700" />
            <prop name="SizeX" value="7620" />
            <prop name="SizeY" value="5978" />
         </model>
         <a661_widget name="TemperatureCelciusScale" type="A661_PICTURE">
            <model>
               <prop name="WidgetIdent" value="0x1222" />
               <prop name="PosX" value="750" />
               <prop name="PosY" value="2390" />
               <prop name="SizeX" value="6126" />
               <prop name="SizeY" value="2490" />
               <prop name="PictureReference"
                     value="SymbolTemperatureCelciusScale" />
            </model>
         </a661_widget>
         <a661_widget name="Symbol1" type="A661_SYMBOL">
            <model>
               <prop name="WidgetIdent" value="0x1223" />
               <prop name="PosX" value="3810" />
               <prop name="PosY" value="2390" />
               <prop name="RotationAngle" value="596523235" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_WHITE" />
               <prop name="SymbolReference" value="VertArrowSymbol" />
            </model>
         </a661_widget>
         <a661_widget name="Symbol2" type="A661_SYMBOL">
            <model>
               <prop name="WidgetIdent" value="0x1224" />
               <prop name="PosX" value="3410" />
               <prop name="PosY" value="2690" />
               <prop name="RotationAngle" value="119304647" />
               <prop name="StyleSet" value="OEM_STYLESET_FREE_COLOR" />
               <prop name="ColorIndex" value="OEM_GREEN4" />
               <prop name="SymbolReference" value="HourGlassSymbol" />
            </model>
         </a661_widget>
     </a661_widget>
   </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-9 Binary DF Specifying Picture Definitions

This section shows a binary Definition File that defines two pictures (bitmaps). See Section C-10 for an XML version of this example.

```
# Binary Definition File
# Example demonstrates how a block of pictures can be defined.

A661            # A661_DF_MAGIC_NUMBER
01              # Library_Version
03              # Supp_Version
6788            # Application Identifier
0000            # UNUSED PAD

# Picture Definitions start here:

BB              # A661_BEGIN_PICTURE_BLOCK
C0              # PixelFormat, value: A661_PIX_FMT_RGBA_8
02              # NumberOfPicturesInBlock, value: 2
00              # NumberOfColorTableEntries, value: 0 (CDS Global Color
Table)
00              # ColorTableFormat, value: 0  (ignore)
000000          # UNUSED PAD

BC              # A661_BEGIN_PICTURE
00              # UNUSED PAD
0200            # picture reference, value: 512
0080            # NumberOfPixelsWidth, value: 128
0080            # NumberOfPixelsHeight, value: 128
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
BD              # A661_END_PICTURE
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
BC              # A661_BEGIN_PICTURE
00              # UNUSED PAD
0201            # picture reference, value: 513
0080            # NumberOfPixelsWidth, value: 128
0080            # NumberOfPixelsHeight, value: 128
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
341A4644        # Pixel Data for Picture
84006125        # Pixel Data for Picture
542B8F00        # Pixel Data for Picture
06E374E1        # Pixel Data for Picture
6C248EF1        # Pixel Data for Picture
00000000        # Pixel Data for Picture
49454E44        # Pixel Data for Picture
AE426082        # Pixel Data for Picture
50011E00        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00006CC6        # Pixel Data for Picture
81690000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
00000000        # Pixel Data for Picture
BD              # A661_END_PICTURE
000000          # UNUSED PAD

00000000        # RESERVED
BE              # A661_END_PICTURE_BLOCK
000000          # UNUSED PAD
# Layer Definition starts here:
A0              # A661_BEGIN_LAYER_BLOCK
05              # layer ID (5)
0000            # context number
0000008C        # block size (140 bytes)

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A1F0            # widget type (A661_PANEL)
1221            # widgetID, value: 0x1221
0000            # parented, (zero indicates layer is parent)
01              # enable, value: A661_TRUE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
0000            # style_set
0000            # UNUSED PAD

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1222            # widgetID, value: 0x1222
1221            # parentID, value: 0x1221
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
0000            # style set
0200            # picture reference, value: 512

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A200            # widget type (A661_PICTURE)
1223            # widgetID, value: 0x1223
1221            # parentID, value: 0x1221
00              # anonymous, value: A661_FALSE
01              # visible, value: A661_TRUE
00002AF8        # pos_x, value: 11000
0000319C        # pos_y, value: 12700
00001000        # size_x, value: 4096
00001000        # size_y, value: 4096
0000            # style set
0201            # picture reference, value: 513

000000          # UNUSED PAD
C0              # A661_END_LAYER_BLOCK
000000          # UNUSED PAD
E0              # A661_DF_FOOTER
000000          # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-10 XML DF Specifying Picture Definitions

This is the XML encoding of the example given in Section C-9. Refer to Section 6.0 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values. The WidgetId values are provided in hex format in this example. Both hex and decimal formats for unsigned integral values are allowed.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE a661_df SYSTEM "a661.dtd">

<a661_df library_version="1" supp_version="3">
   <model>
      <prop name="ApplicationId" value="0x6788"/>
   </model>
   <picturetable>
      <picture_defn name="ScalePicture”>
        <model>
           <prop name="PictureReference" value="512" />
           <prop name="ImageFile" value= "..\pictures\scale2.png" />
           <prop name="PictureFormat" value="A661_PIX_FMT_RGBA_8" />
         </model>
      </picture_defn>
      <picture_defn name="ScaleBackgroundPicture">
        <model>
           <prop name="PictureReference" value="513" />
           <prop name="ImageFile" value=
"..\pictures\scale_background1.png"/>
           <prop name="PictureFormat" value="A661_PIX_FMT_RGBA_8" />
         </model>
      </picture_defn>
   </picturetable>
   <a661_layer name="PICTURE_EXAMPLE_LAYER">
      <model>
         <prop name="LayerId" value="5"/>
         <prop name="ContextNumber" value="0"/>
         <prop name="Height" value="25000"/>
         <prop name="Width" value="20000"/>
      </model>
      <a661_widget name="ExampleScale" type="A661_PANEL">
         <model>
            <prop name="WidgetIdent" value="0x1221" />
            <prop name="PosX" value="11000" />
            <prop name="PosY" value="12700" />
            <prop name="SizeX" value="4096" />
            <prop name="SizeY" value="4096" />
         </model>
         <a661_widget name="Scale" type="A661_PICTURE">
            <model>
               <prop name="WidgetIdent" value="0x1222" />
               <prop name="PosX" value="11000" />
               <prop name="PosY" value="12700" />
               <prop name="SizeX" value="4096" />
               <prop name="SizeY" value="4096" />
               <prop name="PictureReference" value="512" />
            </model>
         </a661_widget>
         <a661_widget name="Pointer" type="A661_PICTURE">
            <model>
               <prop name="WidgetIdent" value="0x1223" />
               <prop name="PosX" value="10000" />
               <prop name="PosY" value="12700" />
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
                <prop name="SizeX" value="4096" />
                <prop name="SizeY" value="4096" />
                <prop name="PictureReference" value="513" />
            </model>
        </a661_widget>
    </a661_widget>
</a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-11 Binary DF Specifying Escape Sequences

This section shows a binary Definition File that includes an example of escape sequences. See Section C-12 for an XML version of this example.

This example shows the text “YOU ARE A BANANA” containing the following escape sequences:

- A foreground color of index 1 is set for the text “ARE A BANANA”
- A background color of index 2 is set for the text “A BANANA”
- The text “ARE” is bold and underlined
- If color index 1 was set to red and color index 2 was sent to yellow, then this text example would appear like this:

YOU **ARE** A BANANA

```
# Binary Definition File
# Example demonstrates how a block of pictures can be defined.

A661            # A661_DF_MAGIC_NUMBER
01              # Library_Version
04              # Supp_Version
6788            # Application Identifier
0000            # UNUSED PAD

# Layer Definition starts here:
A0              # A661_BEGIN_LAYER_BLOCK
01              # layer ID (1)
0000            # context number
0000007C        # block size (124 bytes)

CA01            # A661_CMD_CREATE
0020            # command size (32 bytes)
A1F0            # widget type (A661_PANEL)
1221            # widgetID, value: 0x1221
0000            # parented, (zero indicates layer is parent)
01              # enable, value: A661_TRUE
01              # visible, value: A661_TRUE
00000000        # PosX, value: 0
00000000        # PosY, value: 0
00002710        # SizeX, value: 10000
00002710        # SizeY, value: 10000
0000            # style_set
0000            # UNUSED PAD

CA01            # A661_CMD_CREATE
0050            # command size (80 bytes)
A170            # widget type (A661_LABEL_COMPLEX)
0002            # WidgetIdent, value: 0x0002
0001            # parented, (parent has WidgetIdent 1)
00              # Anonymous, value: A661_FALSE
01              # Visible, value: A661_TRUE
00000371        # PosX, value: 881
00001E58        # PosY, value: 7768
00000ED7        # SizeX, value: 3799
0000032F        # SizeY, value: 815
0000            # StyleSet
0050            # MaxStringLength, value: 80
15              # Alignment, value: A661_CENTER
000000          # UNUSED PAD
1B42001B
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
41001B43
00000000          # DefaultStyleText
594F5520          # LabelString: "YOU "
1B4101            # KforeColor 01
1B4A              # Kbold_B
1B48              # Kunderline_B
415245            # LabelString: "ARE"
1B4B              # Kbold_E
1B49              # Kunderline_E
204120            # LabelString: " A "
1B4202            # KbackColor 02
42414E414E4100    # LabelString: "BANANA"
00                # UNUSED PAD

C0                # A661_END_LAYER_BLOCK
000000            # UNUSED PAD

E0                # A661_DF_FOOTER
000000            # UNUSED PAD
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-12 XML DF Specifying Escape Sequences

This is the XML encoding of the example given in Section C-11. Refer to Section 6.0 for a description of the XML DF format, including the ways that DefaultStyleText can be represented.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a661_df SYSTEM "a661.dtd">
<a661_df library_version="1" supp_version="3">
<model>
     <prop name="ApplicationId" value="1" />
  </model>
  <a661_layer name="testEscaped" >
    <model>
      <prop name="LayerId" value="1" />
      <prop name="ContextNumber" value="0" />
      <prop name="Height" value="10000" />
      <prop name="Width" value="10000" />
    </model>
    <a661_widget name="panel" type="A661_PANEL">
      <model>
        <prop name="WidgetIdent" value="1" />
        <prop name="Enable" value="A661_TRUE" />
        <prop name="Visible" value="A661_TRUE" />
        <prop name="PosX" value="0" />
        <prop name="PosY" value="0" />
        <prop name="SizeX" value="10000" />
        <prop name="SizeY" value="10000" />
        <prop name="StyleSet" value="0" />
      </model>
      <a661_widget name="labelComplex" type="A661_LABEL_COMPLEX">
        <model>
          <prop name="WidgetIdent" value="2" />
          <prop name="Anonymous" value="A661_FALSE" />
          <prop name="Visible" value="A661_TRUE" />
          <prop name="PosX" value="881" />
          <prop name="PosY" value="7768" />
          <prop name="SizeX" value="3799" />
          <prop name="SizeY" value="815" />
          <prop name="StyleSet" value="0" />
          <prop name="MaxStringLength" value="80" />
          <prop name="Alignment" value="A661_CENTER" />
          <structprop name="DefaultStyleText">
            <field name="TOutline" value="0" />
            <field name="TBackColor" value="0" />
            <field name="TForeColor" value="0" />
            <field name="TFont" value="0" />
          </structprop>
          <prop name="LabelString"
value="YOU \x1B\x41\x01\x1B\x4A\x1B\x48ARE\x1B\x49\x1B\x4A \x1B\x42\x02A
BANANA" />
        </model>
      </a661_widget>
    </a661_widget>
  </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

C-13 XML DF Specifying Variable-Structure Properties

This is an example of how to store a StaticParamBufferExtension's BufferArray in XML format.

Refer to Section 6.0 for a description of the XML DF format. In this example not all property values are given explicitly. The unspecified properties take default values.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a661_df SYSTEM "a661.dtd">
<a661_df library_version="1" supp_version="6">
   <model> <!-- properties omitted for brevity --> </model>
   <a661_layer name="Warning">
      <model> <!-- properties omitted for brevity --> </model>
      <a661_widget name="Label" type="A661_LABEL">
         <model> <!-- properties omitted for brevity --> </model>
      </a661_widget>
      <a661_widget name="Button" type="A661_PUSH_BUTTON">
         <model> <!-- properties omitted for brevity --> </model>
      </a661_widget>
      <a661_widget name="BufferFormat" type="A661_BUFFER_FORMAT">
         <model>
            <prop name="WidgetIdent" value="3"/>
            <prop name="NumberOfFields" value="3"/>
            <arrayprop name="BufferStructure">
               <structentry>
                  <field name="WidgetIdent" value="Label"/>
                  <field name="ParameterIdent" value="A661_POS_XY"/>
               </structentry>
               <structentry>
                  <field name="WidgetIdent" value="Label"/>
                  <field name="ParameterIdent" value="A661_STRING"/>
               </structentry>
               <structentry>
                  <field name="WidgetIdent" value="Button"/>
                  <field name="ParameterIdent" value="A661_ENABLE"/>
               </structentry>
            </arrayprop>
         </model>
         <a661_extension type="A661_STATIC_PARAM_BUFFER_EXTENSION">
            <model>
               <prop name="NumberOfBuffers" value="2"/>
               <arrayprop name="BufferArray">
                  <datablock name="Initial Values">
                     <datapair value1="500" value2="2500"/>
                     <datum value="N/A"/>
                     <datum value="A661_FALSE"/>
                  </datablock>
                  <datablock name="Smoke Alarm">
                     <datapair value1="0" value2="3000"/>
                     <datum value="SMOKE ALARM"/>
                     <datum value="A661_TRUE"/>
                  </datablock>
               </arrayprop>
            </model>
         <a661_extension>
      </a661_widget>
   </a661_layer>
</a661_df>
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

This is another example of how to store a StaticParamBufferExtension's BufferArray in XML format. This second case allows changing the LabelString of a Label widget at runtime by setting the index of the extension.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE a661_df SYSTEM "a661.dtd">
<a661_df name="Default" library_version="0" supp_version="6">
  <model>
     <prop name="ApplicationId" value="1" />
  </model>
  <a661_layer name="Default" >
    <model>
      <prop name="LayerId" value="1" />
      <prop name="ContextNumber" value="0" />
      <prop name="Height" value="10000" />
      <prop name="Width" value="10000" />
    </model>
    <a661_widget name="label" type="A661_LABEL">
      <model>
        <prop name="WidgetIdent" value="1" />
        <prop name="Anonymous" value="A661_FALSE" />
        <prop name="Visible" value="A661_TRUE" />
        <prop name="PosX" value="2408" />
        <prop name="PosY" value="3942" />
        <prop name="SizeX" value="3164" />
        <prop name="SizeY" value="1000" />
        <prop name="RotationAngle" value="0.0" />
        <prop name="StyleSet" value="0" />
        <prop name="MaxStringLength" value="20" />
        <prop name="MotionAllowed" value="A661_TRUE" />
        <prop name="Font" value="F2" />
        <prop name="ColorIndex" value="GREEN" />
        <prop name="Alignment" value="A661_CENTER" />
        <prop name="LabelString" value="DEFAULT" />
      </model>
    </a661_widget>
    <a661_widget name="bufferFormat" type="A661_BUFFER_FORMAT">
      <model>
        <prop name="WidgetIdent" value="2" />
        <prop name="NumberOfFields" value="1" />
        <arrayprop name="BufferStructure">
          <structentry>
            <field name="WidgetIdent" value="1" />
            <field name="ParameterIdent" value="A661_STRING" />
          </structentry>
        </arrayprop>
      </model>
      <a661_extension type="A661_STATIC_PARAM_BUFFER_EXTENSION">
        <model>
          <prop name="NumberOfBuffers" value="4" />
          <arrayprop name="BufferArray">
            <datablock>
              <datum value="ONE" />
            </datablock>
            <datablock>
              <datum value="TWO" />
            </datablock>
            <datablock>
              <datum value="THREE" />
            </datablock>
            <datablock>
              <datum value="FOUR" />
```

APPENDIX C
EXAMPLE OF A DEFINITION FILE

```
            </datablock>
          </arrayprop>
        </model>
      </a661_extension>
    </a661_widget>
  </a661_layer>
</a661_df>
```

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

Note: This appendix deleted by Supplement 3.

APPENDIX E MAP MANAGEMENT TUTORIAL

E-1 Examples of Parameters Definition for Map Management

Navigation Display (ND) as the master application

The ND has two basic modes for displaying data. The first one corresponds to Rose or Arc mode where the aircraft representation does not move on the display. The second corresponds to the Plan mode in which the aircraft representation moves and the display is centered on a point which can be very far from the aircraft. Moreover, the ND can represent the data Track-Up, (magnetic or true), Heading-Up (magnetic or true), or North-Up (typical for Plan mode).

FMS as the master application

The master application must first provide a Projection Reference Point (PRP). The PRP will be used by the CDS to know what reference has to be used to run the projection algorithm. Due to the PRP, the CDS will be able to convert latitude/longitude data into a Cartesian coordinate system, for instance, true north oriented and distances in nautical miles.

If the master application provides:

- The position of the PRP on the display
- The orientation of True North relative to the Up direction of the display
- Range information in nautical miles and its correspondence in screen units

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

TCAS as the master application

The TCAS transmits its data as bearing/distance relative to the aircraft with distance to the aircraft and bearing relative to the aircraft main axis. By adding an enumerated value in the “coordinate system” parameter of the MAPHORZ_SOURCE, that means bearing/distance relative to aircraft axis, the CDS will not have enough information to depict the traffic data for TCAS. The CDS needs the following additional data:

- The location of the aircraft on the display
- The orientation of the aircraft relative to the display up direction

There is an aircraft location issue: Aircraft location is set by the master application through the MapHorz. There is an aircraft orientation issue. The master application provides true heading through the MapHorz.

The CDS will implement a bearing/distance MapItem relative to the aircraft. Additional parameters for the MapHorz are:

- Aircraft orientation: relative to True North
- Aircraft latitude
- Aircraft longitude

APPENDIX E
MAP MANAGEMENT TUTORIAL

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 cannot be changed by itself. But because the parameter list of each MapItem is reduced to the useful information only, all the parameters should be set in each SetParameter command.

E-2.1 Addressing MapItems for One Change

Table E-2.1-1 – A661_ParameterStructure for the SetParameter Command for 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
	Latitude
	Longitude
	}
		Parameters of the modified Item.

Note: 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.

APPENDIX E
MAP MANAGEMENT TUTORIAL

Table E-2.2-1 – A661_ParameterStructure for the SetParameter Command

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
	Latitude
	Longitude
	
	20
	0
	A661_LINE_START
	Latitude
	Longitude
	}
		Parameters of the modified items.

Note: The set Parameter command can contain a different type of MapItem.

E-2.3 Removing Map Items

A specific ItemType is A661_NOT_USED. The parameter list for this item would be reduced to the ItemType. This approach declares a previously used Item as not used anymore without faking a type and setting its visibility to HIDE.

Table E-2.3-1 – A661_ParameterStructure for the Set Parameter Command

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-2.4 Drawing Coherent Symbology (Item Groups)

When drawing map data in ARINC 661, items in a MapItem_List can be grouped to create more complex symbology. There are two kinds of item groups:

Symbol Groups

These are used to define Symbols with associated information. These are typically composed of a SYMBOL item containing an EndFlag parameter (some SYMBOLs do not have the EndFlag parameter) and zero, one or more LEGEND items. The EndFlag on the SYMBOL is used by the UA to define whether the text in any LEGEND items following the SYMBOL should be drawn with the SYMBOL. An

example of a Symbol Group is a Flight Plan Waypoint Symbol with Text Identifier, represented in a MIL buffer by a SYMBOL_GENERIC followed by one or more LEGENDs.

Line and Polygon Groups

These are used to draw coherent line sequences (Line Item Group) and polygons (Polygon group). These item groups are typically composed of a LINE_START item followed by one or more LINE_ARC(s) and/or LINE_SEGMENT(s), or a FILLED_POLY_START followed by one or more LINE_SEGMENT(s). The points that make up the Line or Polygon Group are drawn continuously until an item is encountered that has the EndFlag set to TRUE. For example, a Flight Plan Path can be constructed in a MIL buffer as a LINE_START item followed by LINE_ARCs and LINE_SEGMENTs.

For both kinds of item groups, the value of the EndFlag parameter indicates whether or not the end of the item group has been reached. In general, an EndFlag set to TRUE delineates the end of an item group.

When a UA builds a map block (MIL buffer) it may not always want to order map objects by type, i.e., draw all lines first, then symbols and then legends. More typically lines, symbols, and legends are intermixed and grouped functionally within the block to achieve a desired effect. This complicates the use of the EndFlag and as a result some additional clarification is required on its use under these conditions.

The EndFlags for Symbol Groups and Line Groups should be managed independently. As a result, ending one type of group should not affect the ending of the other type, even if the map items in the two groups are interspersed within the map block.

Symbol Groups

For the first kind of item group, SYMBOL items containing the EndFlag parameter can be drawn with associated LEGEND item(s). In this case, the SYMBOL item’s EndFlag parameter is set to FALSE. The following LEGEND item(s) will be interpreted as part of the Symbol Group until the first LEGEND item with its EndFlag parameter set to TRUE is encountered.

The set of items from the first SYMBOL item (with EndFlag = FALSE) to the last LEGEND item (with EndFlag = TRUE) is a complete item group.

Associated error detection and handling (e.g., no EndFlag = TRUE is present) is implementation dependent.

An item that does not contain an EndFlag parameter (e.g., SYMBOL_OVAL) will be interpreted as being a complete Symbol (a complete Symbol Group).

APPENDIX E
MAP MANAGEMENT TUTORIAL

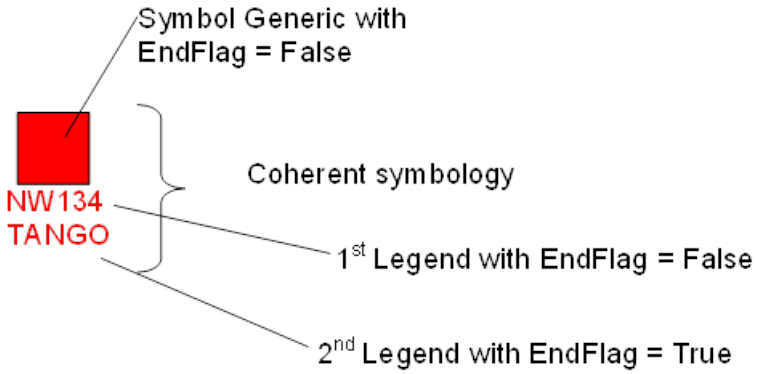


Figure E-2.4-1 – Example of a Symbol Group

Line and Polygon Groups

For the second kind of item group, the first item is a LINE_START or FILLED_POLY_START.

For Line Item groups, the following items will be LINE_SEGMENT or LINE_ARC items with EndFlag = FALSE. The last item of the item group will end with a LINE_SEGMENT or LINE_ARC item with EndFlag = TRUE. This set of items constitutes a complete Line Item group.

For Polygon Item groups, the following items will be LINE_SEGMENT items with EndFlag = FALSE. The last item of the item group will end with a LINE_SEGMENT item with EndFlag = TRUE. This set of items constitutes a complete Polygon Item group.

Associated error detection and handling (e.g., no EndFlag = TRUE is present) is implementation dependent.

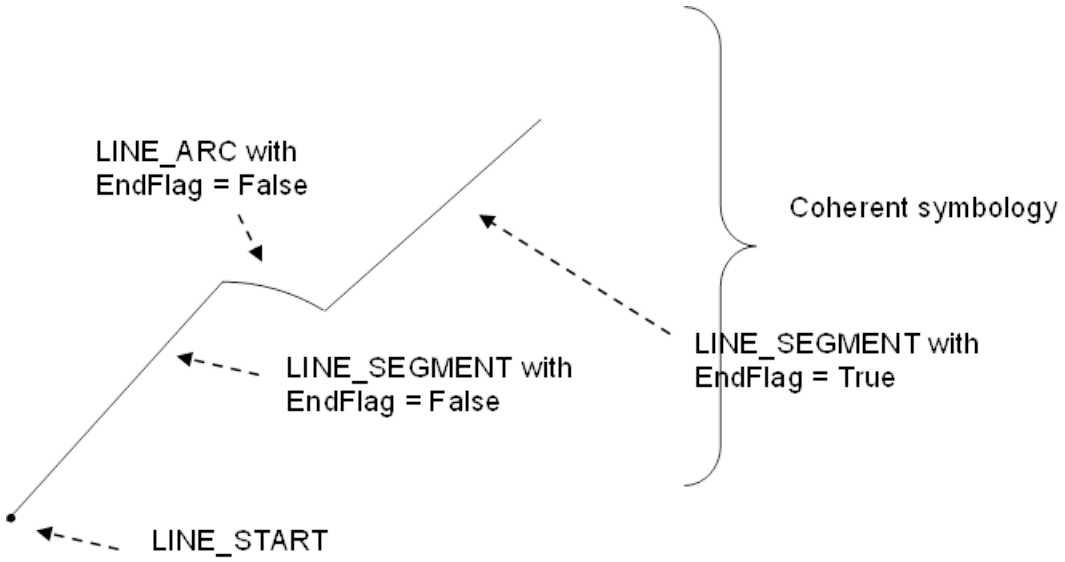


Figure E-2.4-2 – Example of a Line Item Group

APPENDIX E
MAP MANAGEMENT TUTORIAL

Embedding Symbol Groups in Line Groups

The CDS should allow Symbol Groups to be inserted between the start item of a Line Group (LINE_START) and the line closure item (A661_LINE_XXX with EndFlag = A661_TRUE). When a Symbol Group is inserted inside a Line Group, it should be closed (by an item having an EndFlag attribute set to TRUE) before the Line Group definition continues.

Note: This does not apply to Polygon Groups (i.e., a group starting with item POLY_START): they should not be interrupted by Symbol Groups.

Associated error detection and handling (e.g., a Symbol Group inside a Line Group that is not terminated by an EndFlag = TRUE) is implementation dependent.

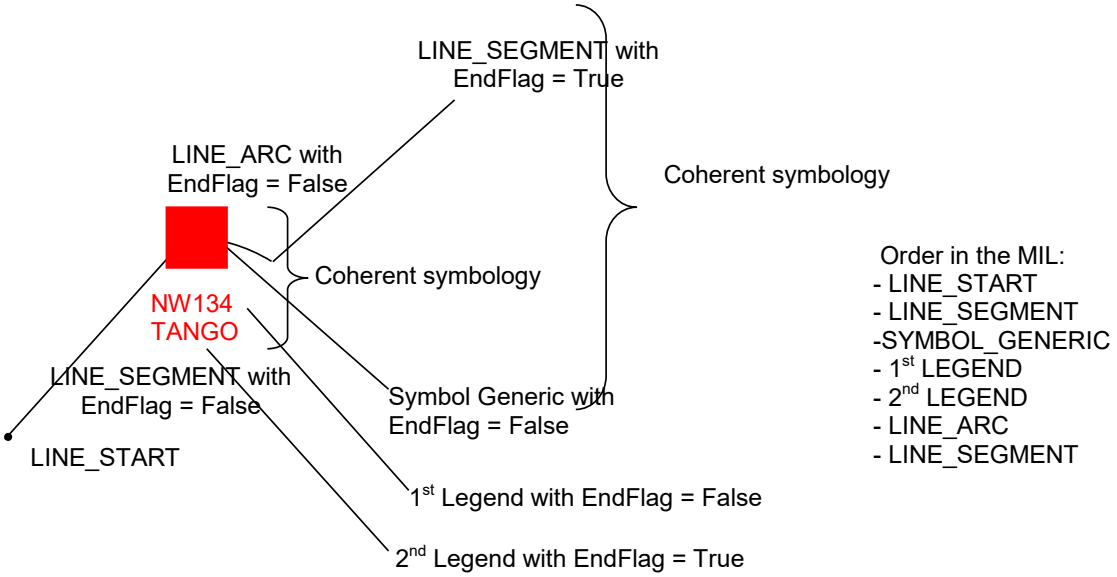


Figure E-2.4-3 – Example of a Symbol Embedded in a Line Item Group

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 meantime, the FMS has changed the information associated with "Widget 201, Item 32," which now carries "NEW YORK".

The problem is that the FMS cannot decide what the event truly means: has the pilot has selected PARIS or NEW YORK?

To address this problem, the FMS could have several solutions. One solution is to manage the Context Number. The UA can change the Context Number by changing the functional information attached to a widget or simply to an Item. In this way, the FMS will have the knowledge of the selected waypoint by correlation between the Context Number and the ident of the selected waypoint.

E-4 Dynamic Priority Management Inside MapHorz_ItemList

The Items inside a MapHorz_ItemList are defined at run-time. The order of the Item inside the MapHorz_ItemList defines the drawing order of the items defined by the ItemIndex parameter of the item. The Item with the highest ItemIndex has the highest drawing priority. Figure E-4.1 illustrates an example of dynamic priority management.

The UA induces a specific drawing order of symbology by ordering the item inside a MapHorz_ItemList. If the UA does not specify the drawing order, then the drawing order should be defined statically in the DF with different MapHorz_ItemList for the set of symbols and drawing orders. Nevertheless, this different set of symbols corresponds to different functional sets, for which it should define a different widget (MapHorz_ItemList).

However, the drawing priority of an Item is defined by the ItemIndex of the Item. Nevertheless, the ItemIndex of an Item is independent of the transmittal order of the Item in the command. In Figure E-4.1, the UA, for instance the FMS, may have to respect a transmitting order. But the transmitting order is independent of the order of the ItemIndex declaration inside the SetParameter command.

APPENDIX E
MAP MANAGEMENT TUTORIAL

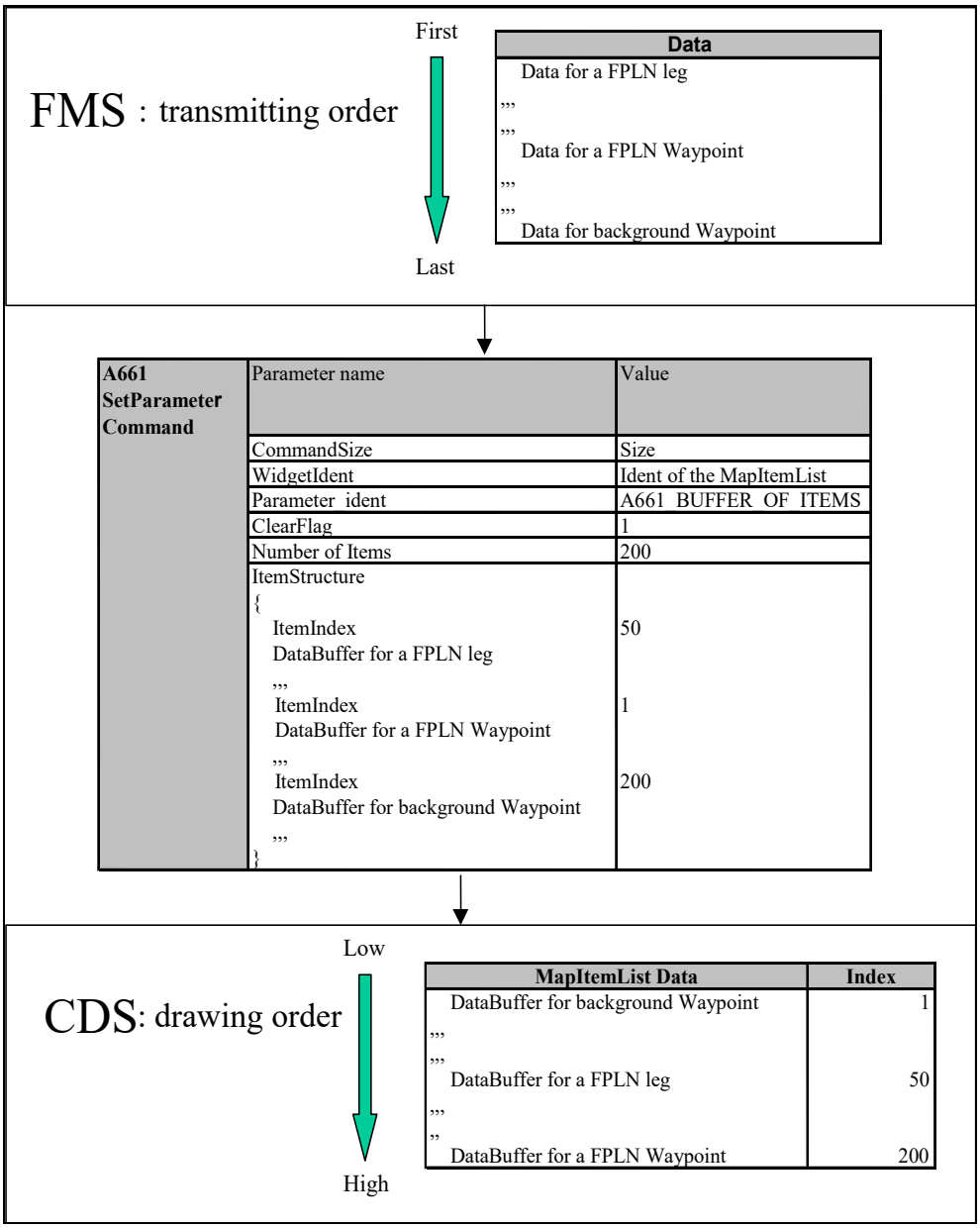


Figure E-4.1 – Example of Dynamic Priority Management

APPENDIX F COMMUNICATION TRANSPORT PROTOCOLS

F-1 Introduction

ARINC 661 defines data communication in the run-time format exchanged between User Applications (UAs) and the CDS. However, it does not specify the transport mechanism for such run-time messages. This was deliberately excluded in early versions of ARINC 661 because the computing environment and networking infrastructures used in aircraft cover such a broad range – from ARINC 429 interfaces to Ethernet derivatives, from simple real-time executives to partitioned operating systems with inter-partition communications mechanisms. Specifying exactly how ARINC 661 run-time messages are delivered to a receiver seemed to be too restrictive. Thus, this detail remains as an exercise for the implementer and the aircraft system integrator.

With this background, this appendix provides an overview of how ARINC 661 communications can work in several typical environments. It makes suggestions on how to deal with specific problems that may arise. It is expected that the information in this appendix will be helpful to system architects who want to consider the use of ARINC 661. Following the scenarios and suggestions described herein is not required to achieve compliance with ARINC 661

F-2 ARINC 661 in Typical Aircraft Environments

Before providing specific examples and solutions, this section provides a brief summary of communication options that ARINC 661 display system designers may wish to consider.

F-2.1 Local/External Applications

This appendix describes communication transport protocols. It assumes the CDS and the UA are not located in the same Line Replaceable Unit (LRU). Thus, the UA is considered an external application.

ARINC 661 also allows UA and CDS to be located in the same LRU. In this case, ARINC 661 run-time messages are exchanged only within this one LRU. This approach may be chosen for design reasons (e.g., reuse of existing ARINC 661 application code, use of ARINC 661-based modeling or other development tools) or to provide for future growth that allows external UAs to use the CDS.

This case will likely have an implication for systems design: a communications mechanism between the UA and the CDS components that provides both high integrity and high reliability exists within this LRU. The following sections do not address architecture-dependent communications inside of an LRU.

F-2.2 Interactivity Impact on Communication

Interactivity in flight deck formats means that bi-directional communications between the CDS and the interactive UAs is required, such that widget events can be communicated to the CDS. While the communications link may be bi-directional in many cases, even for applications that do not display interactive formats, systems in which a UA “blindly” transmits run-time messages to the CDS are certainly possible. Other CDS-generated messages (such as error messages, notifications, etc.) can then not be sent to the UA, but that may be acceptable under the right circumstances.

F-2.3 Communication Type

Communication between CDS and UA can be periodic or event based:

- Data sent by a UA (periodic)

  Traditionally, display systems have received periodic updates (at various rates) of all aircraft data needed for the display. While data updates will likely continue to be periodic for many applications, for the communications link, having an application send periodic updates of all appropriate widget parameters means that the loss of an individual run-time message every now and then due to communications problems is acceptable.

- Data sent upon event (non-periodic)

  ARINC 661 gives UAs the option to send widget parameters only when values have actually changed. Doing this leads to software that resembles PC-based software applications more than the traditional avionics display software; however, there may be cases where that is appropriate, especially for interactive, “PC like” applications that are ported to an ARINC 661 display system.

  If UAs base the generation of some or all of their run-time messages on changed values, then the run-time messages must be sent such that they will arrive intact at the receiver with a very high probability, and a transport mechanism that can guarantee the safe delivery of information must be chosen.

  Depending on how UAs and the CDS communicate, providing that type of transport mechanism can be a non-trivial task. This may lead designers to quickly dismiss the idea of sending any widget parameters only when data changes, and instead send all necessary parameters periodically, all the time. This, however, does not completely solve the problem, because many UAs require a guaranteed delivery of widget events, notifications and other CDS-generated messages that, due to their nature, cannot just be sent to the UA many times. Once this problem is solved, the solution can be applied to messages traveling the other way, too, meaning that UAs can use the same mechanism that guarantees delivery of event messages for widget parameter updates that are not periodic.

- Hybrid designs

  Many hybrid designs are possible: displays that are driven by both ARINC 661 applications and the traditional ways (for different formats or different features within a format); display LRUs that contain some of the UAs but have network interfaces to communicate with others; some UAs sending ARINC 661 run-time messages blindly to a CDS while other UAs maintain bi-directional communications with the CDS, to name just a few of the possibilities. ARINC 661 does not prevent (or even discourage) to mix and combine different solutions where that makes sense.

The following sections show how ARINC 661 can work in selected, typical environments. Detailed discussions of industry terms (such as Ethernet) and other ARINC specifications are not provided here; there is no lack of other information about these terms and concepts for the interested reader.

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS

F-3 Ensuring Reliability

For the purposes of this discussion, the term “reliability” refers to the probability that a message sent by some transmitting application is received by the receiving application. Typical network connections (such as Ethernet, ARINC 429, etc.) drop data periodically, so they are not reliable unless some mechanism is added at a higher level to deal with dropped data. This is not to be confused with network integrity, which describes the probability that a message that arrives at the receiving application has not been accidentally corrupted (without detecting the error) during the transmission. In other words, high integrity ensures accuracy of received message, while high reliability ensures complete receipt of all sent information.

Data connections used in avionics systems typically have good integrity but relatively poor reliability. There are three different ways to deal with the reliability issues:

- Periodic Transmissions:

  Parameters are sent repeatedly at some periodic rate. Examples where this is typically done include aircraft attitude, airspeed, and altitude data. The loss of a single piece of data is not significant because the transmitter will send new values anyway. The probability that two, three, or more subsequent instances of a data value are all lost is much smaller than the probability to lose a single value, and for this reason, low network reliability is typically not an issue if periodic transmissions are used.
- Acknowledgements (ACK/NACK):

  Acknowledgement-based protocols require that the receiver sends a confirmation to the sender for information that it has received. That way, the loss of a data packet is recognized and the sender can send another copy of the original data over the network. These protocols exist in various different forms. Some widely known examples are TCP and TFTP.
- Sequence Numbers:

  Another mechanism to detect the loss of data packets is to have the receiver add a sequence number to each packet, which the receiver monitors. The receiver expects that subsequent messages have adjacent sequence numbers; if it receives anything else, it can assume that a message has been dropped, and some corrective action can be initiated.

  Note: These concepts are not mutually exclusive.

F-4 Ethernet

Ethernet and UDP-based network connections (such as ARINC 664 Ethernet) are generally a good choice for ARINC 661, because they offer a high bandwidth along with good integrity. They are also bi-directional. However, Ethernet is not, by itself, reliable, so if a guaranteed delivery of run-time messages is required, an additional protocol level above UDP or IP is required.

Several industry standards exist that make Ethernet more reliable. The two most widely used ones are TCP and TFTP. Both TCP and TFTP require the receiver to send a confirmation for received messages. If a confirmation is not received back at the sender within a certain amount of time, the sender will retransmit the information.

Both TCP and TFTP can be useful to ensure reliability for ARINC 661 run-time messages. A few things should be considered carefully, though:

- With TCP, sender and receiver determine the size of a “window”, and the sender is allowed to send as much information as will fit in this window until a receipt for a portion or all of the window’s data is received. Thus, TCP does not generally block the sending application – it only does so if the entire window lacks an acknowledgement through the receiver.

  Advantages:

  - Transmit run-time message reliably over an unreliable network

  Drawbacks:

  - TCP is a very complex protocol. It was created to solve transport problems in the world-wide Internet, and many of those problems do not really exist on aircraft networks. If provided on an aircraft network, TCP can be used for ARINC 661, but its complexity and overhead should be kept in mind.
- In TFTP, each message sent by a transmitting application must be confirmed by the sender before more information can be sent. In case of a communication problem, TFTP will prevent a sender from sending new messages until the problem has been recovered by a re-transmission. In case of re-transmission failure, a high level communication sanction has to be taken, such as Communication re-initialization.

  Advantages:

  - Transmits run-time message reliably over an unreliable network
  - It allows latency and network bandwidth optimization (by sending only once the message).
  - This communication protocol is well-known (and thus the associated drawback effect).

  Drawbacks:

  - Blocking nature of the TFTP, UA application has to receive the message acknowledge before sending new data. Thus, particular attention should be paid to TFTP and network configuration. Besides, re-transmission could lead to additional latency in degraded cases.
  - For connections that carry periodic data, this is probably a poor choice, whereas for certain interactive, “PC like” formats it can work well.

F-5 ARINC 429

ARINC 429 is a serial interface that is widely used for exchanging information on aircraft, including for display systems. It is unidirectional and does not provide a guarantee for delivery of sent messages. Its wide availability on aircraft makes it an interesting consideration for ARINC 661, though.

ARINC 661 run-time messages can be sent over ARINC 429 connections. The result is somewhat similar to Ethernet/UDP, operating at much slower data rates, though. If reliable communications is needed, an intermediate protocol between ARINC 429 and the ARINC 661 run-time messages can be used, similar to TFTP and TCP (as discussed above).

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS

F-6 Example Protocol for Reliable Communications

This reliable communications protocol is based on sequence number. It is suggested as an optional mechanism for ARINC 661 display systems, and can serve as a starting point when circumstances require the use of such a mechanism.

The main goals for the design of the reliable communications mechanism are:

- Transmit run-time message reliably over an unreliable network
- Determinism – achieve predictable network load and behavior
- Include status (health, availability) information into the design
- Compatible with multicast/broadcast transmission

The protocol works mostly by sending out multiple copies (spread out over time) of data that a sender considers important, whereas other data is sent only once, such as data that is generated cyclically anyway and for which a lost value every now and then would not cause a problem.

Drawback effect of this proposal could be:

- Bandwidth use of the network: Need to transmit several times a set of data depending of network reliability.
- Next message is blocked until all the multiple copies of the previous message have been sent.

The mechanism consists of two levels of wrappers around ARINC 661 run-time data: a “group” and a “block.” Conceptually, a “block” can be thought of something that an application transmits (at least) one of per cycle. It is atomically transmitted by the network in that it is either received and undamaged, or not visible to the receiver at all. A block consists of the block header and zero or more “groups.”

A “group” consists of one or more ARINC 661 run-time message blocks. Anytime a UA or a CDS intends to send an ARINC 661 run-time message block, that block is wrapped in a “group.” “Groups” can be sent reliably or unreliably, which is a choice the sender makes.

The sender assigns a sequence number to each group. Groups that carry unreliable data have a sequence number of zero. Groups that carry reliable data have a sequence number greater than zero. The sender starts with a sequence number of one for the first reliable group, and increments it by one each time a new reliable group is generated. Copies of a reliable group have the same sequence number as the first version of that group.

The Groups within an Extended Block are assembled so that they are in order of sequence number, from lowest to highest, except that groups with a sequence number of zero (i.e., unreliable groups) may appear anywhere within an Extended Block.

APPENDIX F
COMMUNICATION TRANSPORT PROTOCOLS

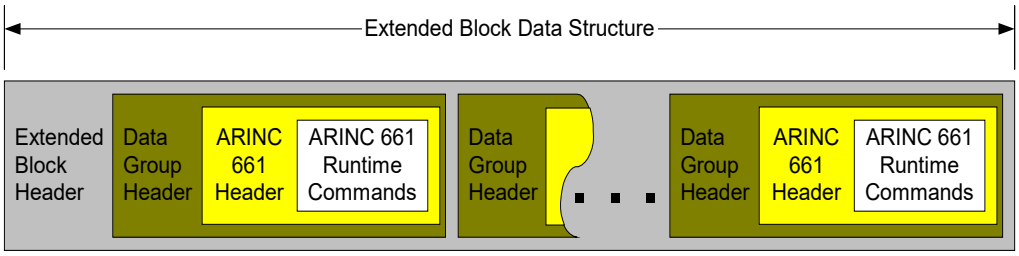


Figure F-6 – Extended Block Data Structure

Extended Block

The Extended Block is structured as defined in Table F-6.1.

Note: The first 12 bytes are the Extended Block Header.

Table F-6.1 – Extended Block Structure

Offset (bytes)	Size (bits)	Parameter	Description	Value/ Type
0	16	Start Marker	Indicates the start of the Block Header.	"BS" 0x4253
2	16	Extended Block Size	Size in bytes of the extended block including the header.	0-65535
4	4	Source	Identifies the CDS or UA originator of the message block.	0-15
	4	Destination	Identifies the CDS or UA to which the message block is destined.	0-15
5	8	Number of Groups	Number of groups in this extended block.	0-255
6	8	Service Available	Used by UAs to indicate whether the service associated with this connection is available for use by the CDS.	True/ False
7	8	Assumed Health	For each connection, the CDS and UAs compute health of their counterpart and echo this assumed health in subsequent transmissions.	True/ False
8	32	Lowest Sequence No.	Used by the reliable communications protocol to indicate the lowest sequence number of data that is being or will be sent.	
12	{32}+	Extended Block Data	The extended block data is made up of any number of Data Groups.	

The values for "Source" and "Destination" are meant for multi-cast scenarios, where more than one UA or CDS listens on any given connection. In this case, a number between 0 and 15 is assigned to each sender and each receiver on each connection. These fields can be ignored if point-to-point connections are used between each CDS and each UA.

Both UAs and the CDS exchange extended blocks periodically to exchange health information. If a UA or a CDS determines that its own health is bad, it should stop sending periodic extended blocks. Extended blocks may contain any number of groups, including none if there is no actual data to send and the purpose of the extended block is merely to announce the sender's health.

The intended use for the "Service Available" parameter means that it should be set either by a CDS or a UA. The idea is that either a CDS picks a UA to drive a layer, or a UA picks a CDS to display its layers. In either case, the "Service Available" flag is set to True if the sender is able to support a new request from the receiver.

The "Assumed Health" parameter is used to tell the receiver what health it assumes for that receiver. For example, if a CDS is communicating to a UA, the CDS will determine the health status of that UA (either Failed or Valid). When the CDS transmits data to UA it will set the value of "Assumed Health" in the Extended Block Header to the value of the Health Status it computed for the "UA". This helps address problems that might otherwise arise from a failure in just one way of a bi-directional connection.

The use of the Lowest Sequence Number parameter is defined below (see "Reliable Communications Method").

Data Groups

The Data Group is structured as defined in Table F-6.2 below.

Note: The first 8 bytes are the Group Header.

Table F-6.2 – Data Group Structure

Offset	Size	Parameter	Description	Value
0	16	Start Marker	Indicates the start of the Data Group Header.	"GS" 0x4753
2	16	Data Group Size	Size of the group in bytes (incl. the group header).	
4	32	Sequence Number	Sequence Number for the group.	
8	{32}+	Group Data Block	Contains one or more ARINC 661 data blocks and a maximum of one control block.	

The Sequence Number is defined in the following section.

Reliable Communications Method

The reliable communication method allows a CDS or UA to transmit a message block multiple times and allow the receiving system a means of determining whether it has missed one of these blocks.

Note: The number of times the transmitter sends the block is variable and can be configured to achieve the desired reliability across the network. The assertion being that sending the block multiple times increases the probability that one of the blocks will arrive.

Transmitting System

When a connection is first established, the transmitting system sets the value of Transmitted Sequence Number to zero. Each time the sender creates a fresh block of new data that needs to be transmitted it increments the Transmitted Sequence Number by one and sets the Sequence Number in the Group Header to this value.

Note: If a group is to be sent multiple times, then each duplicate message will have the same sequence number.

If the sender wishes to send a non-reliable block of data it sets the Sequence Number in the Group Header to zero.

The sender sets the value of Lowest Transmitted Sequence Number to the lowest non-zero sequence number for a message block that is still scheduled for re-sending, or the next future sequence number if there is nothing more to send.

The sender transmits this group in successive message blocks until it has been sent the number of times required to assure reliable transmission.

Receiver

When a connection is first established, the receiver sets the value of Expected Next Sequence Number to one.

The receiver compares the value of Expected Next Sequence Number to the Sequence Number value in each Group Header for new message blocks. If the Sequence Number in the Group Header is equal to the Expected Next Sequence Number then the receiver processes the associated block of ARINC 661 data, increments the Expected Next Sequence Number by one and then process the next group.

If the Sequence Number in the Group Header is not equal to the Expected Next Sequence Number, then the receiver skips to the next group.

If the Lowest Sequence Number in the Extended Header is higher than the Expected Next Sequence, the receiver knows that it has experienced an unrecoverable loss of data. To resynchronize, the receiver then sets the Expected Next Sequence Number to the Lowest Sequence Number in the Extended Header and then asserts the “Data Lost” state. If the receiver is a CDS, it sends the A661_NOTE_REINIT_LAYER to the associated application. If the receiver is a UA, it reinitializes all layer data as if it had received an A661_NOTE_REINIT_LAYER message. And the data is recovered.

APPENDIX G NEW WIDGET GUIDELINES

G-1 Guideline Purpose

The purpose of New Widget Guidelines is to:

Assist candidate widget advocates in successfully proposing new widgets.

- Assist reviewers in a conducting a structured evaluation of proposed new widgets.

The proposal of new widgets can be divided into two general parts.

1. The Description / Discussion section that provides a higher level discussion of the proposed widget's purpose.
2. The Template section that provides the detailed widget attributes that will ultimately be included in ARINC Specification 661.

G-2 Description/Discussion

Potential new widgets should be introduced with a high level discussion that addresses why the proposed widget should be added to the Widget Library. As applicable, that discussion should include the following points:

Describe the new widget's purpose and the operational capability it provides for the aircraft. If possible, provide a representative figure and illustrative example. If applicable, describe how the user interface functions. Where does the new widget fit in the A661 library structure?

- If applicable, explain why the current library is insufficient to accomplish the requirement. What does the candidate widget do that cannot be reasonably accomplished with the current widget library?
- Is this new widget an extension of a current capability or a fundamentally new capability?
- If the proposed new widget extends a current capability, it might be better to add a Widget Extension instead of a new widget. See Section 8.2, Rules for Extensions.
- How does this widget interface with other widgets?
- Could a current widget be modified to accomplish the intended purpose? If so, why is a new widget a better idea? How does this proposed widget differ from related widgets (what attributes are different)? Why is the new widget better than a previous widget?
- Are there any insurmountable backward compatibility issues with this widget or widget change?
- Does this widget make any current widgets obsolete and thus eligible for removal from the standard? If so, decide which is the best way to remove it (see the paragraph below about how widgets can be removed from the standard).
- Does the new widget design include any growth capability?
- Are there likely to be any processing or bandwidth resource issues with the new widget?

- Is the definition of widget parameters independent of “look and feel” concerns that could be better addressed by StyleSet?
- Is the new widget best suited as a standard ARINC 661 library object or is it more appropriately implemented as an OEM custom widget?
- If the introduction of the new widget introduces new concepts beyond those already defined in ARINC 661, the advocate should include material that describes the advantage of the proposal and how it might be specified in ARINC 661.

There are at least two ways to remove a widget from this standard. In many cases widgets will simply be deleted from ARINC 661 when they are deemed to be no longer useful. The second way to remove a widget from ARINC 661 is to mark the widget described in the document as “Deprecated.” This is a phased removal allowing changes to be made to ARINC 661 without causing earlier implementations to become immediately obsolete. A statement such as “This section was Deprecated by Supplement X” will be inserted in the document. This will indicate that the widget (or widget feature, etc.) is out of date and will be removed from the document by a later Supplement. Often a new widget will be added to replace the Deprecated widget, but it should be borne in mind that equipment suppliers will be expected to support deprecated features as long as they remain in ARINC 661.

G-3 Widget Design Guidelines

The following guidelines should be considered when designing a new widget. If a widget does not comply with these guidelines, an explanation should be provided.

Each rule is associated with an example to illustrate the way it should be understood (example of good usage of the rule or of violation of the rule).

1. Each widget should have a single role / responsibility. If a widget has several roles (for instance depending on a parameter), it should be split into a set of simpler widgets, each having a single role.

   E.g., a widget could be a scrolled-list or a pop-up list depending on its size (still allowing to select one item out of several), but a widget cannot be a pop-up list (selecting an item out of several) and a text-field (entering a text) depending on the value of its parameters.
2. Each parameter should have a single purpose. If a parameter has several purposes, it should be split into a set of simpler parameters each of which has a single purpose. Further, there should be no dependencies between parameters.

   Example of rule violation: ExternalSource SourceDX and SourceDY parameter have different behavior depending on the Source parameter.
3. In general, a widget’s design should not be limited by or depend upon assumptions regarding the types of widgets it can be contained by.

   Example of rule violation: the TabbedPanel widget can only be the child of a TabbedPanelGroup widget.
4. Widget parameters should be individually modifiable: provided a parameter value is set consistently with its usage domain, setting this value should not result into an erroneous behavior due to the value of other parameters.

Example of rule violation: the AlphaMask and NumericMask values cannot be changed if not consistent with the labelString.

5. Each layer has an associated namespace (widget ID). The parameter of a widget belonging to a layer should not make reference to the namespace of another layer (in this case, use a CDS-level reference).

   Example of rule violation: FocusLink is making a reference to a widget belonging to another layer.

6. StyleSet should not be used to describe anything but alternative views. StyleSet should not change the function of a widget.

   Example of rule violation: the StyleSet should not be used to change one widget's appearance or function into that of another.

7. No widget should rely on assumptions about the cockpit configuration (number of users, number of cursors, size and number of displays, type of control, etc.).
8. All parameters should be definition and run-time modifiable except if this would raise a safety or system issue. This analysis should be done before adding a new widget to ARINC 661.

   Example: Maximum string length is not run-time modifiable because changing the value of this parameter at run-time may result in undesired effects (memory allocation issue).

9. If a widget is derived from an existing widget, it should maintain structural commonality with that widget (where practical) for those aspects that are functionally the same.

   Example: PicturePushButton is derived from the PushButton.

10. All widgets should follow the established conventions regarding parameter types/sizes/run-time identifiers/events. For example, all parameters related to screen dimension should be 32 bits and should be expressed in units of 1/100th millimeter.
11. The naming of widget parameters should be consistent with how existing widget parameters are named. Parameters should be spelled the same as in existing widgets. Spaces should not appear in parameter names (unless required for document formatting purposes).

G-4 New Widget Template

Section 3.3 describes ARINC 661 widget characteristics and the interface.

Each widget definition is divided into the following parts that need to be included in a new widget proposal.

1. Definition Section
   a. Categories
   b. Description
   c. Restriction
2. Widget Parameters Table
3. Creation Structure Table
4. Event Structure Table
5. Run-time Modifiable Parameter Tables

While it may not be necessary to complete the entire widget definition when initially proposing a widget it is helpful to highlight the areas that make the widget unique.

Definition Section

G-5 Widget Category

Widgets are grouped into one or more categories based on the widget's purpose. Table 3.2.2-1 is the current list of Widget Library Categories. The following is provided for convenience.

- Container
- Graphical Representation
- Text String
- Interactive
- Map Management
- Utility
- UA Validation

Description

Provide a functional description of the widget.

List any widgets it is similar to or dependent upon.

Restrictions

Describe any restrictions to ARINC 661 principles.

Describe any exceptions to defined provisions. For example, Sections 2.3.4.1 through 2.3.4.3 provide defined conventions for positioning and sizing within a window. Exceptions to these conventions should be identified and explained.

Widget Parameters

The parameters associated with each widget can be divided into commonly used parameters and specific parameters. Specific parameters can be either previously used or new ones. Requirements for new specific parameters will need to be clearly explained as part of the new widget proposal.

The Commonly Used Parameters List below is provided for convenience and lists common parameters and identifies the parameters that are required for all widgets. The list is an abbreviated compilation of the information found in Section 3.1.3.

Section 3.1.3 provides a full discussion of Commonly Used Parameters.

Pay special attention to the StyleSet parameter. It is desirable for the ARINC 661 interface to be as independent as possible from the widget's graphical characteristics. The parameter allows the different OEMs to ensure the widget graphics meets their requirements.

The following table is provided to assist in listing the new widget's parameters.

APPENDIX G
NEW WIDGET GUIDELINES

Table G-5 – Widget Parameter Table Template

Parameters	Change	Description
Commonly Used Parameters		
		See Commonly Used Parameter list below or Section 3.1.3
Previously Used Specific Parameters		
		See Table 4.6-8 – Parameter Types
New Specific Parameters		

G-6 Commonly Used Parameters List

Note: (Req) indicates a required widget parameter

Table G-6 – Commonly Used Widget Parameters

Widget Parameter	Category Description
WidgetType (Req)	Type of widget See Table 4.6-7
WidgetIdent (Req)	Identifier of the widget (refer to Section 3.1.1,) WidgetIdent is a non-null positive value. NULL is reserved for referring to the layer level (e.g., ParentIdent)
ParentIdent (Req)	Identifier of the immediate container of the widget. Only a special category of widgets called “Container” can be the parent of other widgets. At the highest level of the widget hierarchy within a layer, the ParentIdent value is 0 (NULL). This means that the parent of the widget is the layer.
Visible	A661_FALSE: The widget will not be rendered. A661_TRUE: If all its ancestors are visible, the widget will be rendered. If one or all its ancestors are invisible, the widget will not be rendered, whatever the value of its visible parameter.
Enable	A661_FALSE: The widget will not be interactive. A661_TRUE or A661_TRUE_WITH_VALIDATION: If all its ancestors are enabled, the widget will be interactive. If one or all its ancestors are disabled, the widget will not be interactive, whatever the value of its Enable parameter. An invisible widget is not interactive, independent of the value of its Enable parameter.
Anonymous	A661_FALSE: run-time accessible. Widget can be modified at run-time, if it has some run-time accessible parameters. A661_TRUE: anonymous. Widget cannot be modified at run-time by UA. CDS behavior when a UA attempts to SetParameter on an anonymous widget is undefined.
StyleSet	StyleSet allows the UA to select from a predefined set of graphical characteristics to be applied to a widget. See Table 3.1.3.3 for a full discussion of the StyleSet parameter.
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.

APPENDIX G
NEW WIDGET GUIDELINES

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

G-7 Creation Structure

A Creation Structure Table should be prepared for each candidate widget.

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

The widget parameter order discussed in the previous section may be different from the order in the widget parameter table. The widget parameter table describes parameter functional aspect, while the creation structure table describes the parameter buffer coding aspect.

Some points to remember when generating a new Creation Structure:

- If possible, allocate a set of pad bits (16 to 32 bits) at the end of the widget creation structure for future growth.
- Ensure the StyleSet size is 16 bits.
- The Enable flag goes just before Visible flag (exceptions are the Connector, DataConnector, and CursorOver widget).
- Use existing parameter types as much as possible, refer to tables in Section 4.
- When adding codes to the tables in Section 4, be careful to avoid adding a code to a table that has already been assigned in that table (exception: constants can be duplicated). Some checking beforehand is required before assigning a code: not all tables are ordered according to increasing code value, some are ordered alphabetically or per some other method.
- Use {n}+ notation to indicate a variable sized buffer of parameters.
- Pads are all type N/A. Their value should be set to zero (0).
- Within a word, place 16-bit parameters before 8-bit parameters.
- Add pads at the end of any unfilled 32-bit words.

- When creating a new indexable array (StringArray, EntryArray, etc.), start the array indexing with the number one (1) to be consistent with this specification. Note that an index value of zero may have special meaning, as it does for ComboBox and ScrollList. Any exception that must be made to this rule should be clearly noted in the widget description.

The following table is provided to assist in completing the new widget's Creation Structure table.

Table G-7 – Creation Structure Table Template

CreateParameterBuffer	Type	Size (bits)	Value / Range When Necessary
WidgetType	ushort	16	See Table 4.6-9, Widget Types
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
			Continue the table, ensuring all widget parameters are included.

G-8 Event Structure

Interactive widgets must have an event structure table attached. This structure is similar to the widget creation structure and provides the structure for widget associated events.

An Event Structure Table should be prepared for each Event.

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

A discussion of widget events can be found in Section 3.1.4, Widget Events.

Table 4.5.4.2-3 defines the run-time Widget_Event_Structure.

The following table is provided to assist in completing the new widget's Event Structure Table.

Table G-8 – Event Structure Table Template

Event Structure	Type	Size (bits)	Value/Description
EventIdent	ushort	16	See Table 4.6-9, Event Types
As required.			See Table 4.6-10, Boolean Constants
			See Table 4.6-11, Integer Constants
UnusedPad	N/A	variable	Set value to "0"

G-9 Runtime Modifiable Parameters

The dynamic data transfer between the UA and CDS at run-time includes the requirement to update run-time widget parameters. Runtime Modifiable Parameters should include all widget parameters that have "R" in the Widget Parameter Table Change column.

Section 4.5.4.5, ARINC 661 Parameter Structure, provides details of the parameter structures which should be applied to run-time modifiable parameters.

Some points to remember when generating a new runtime modifiable parameter:

APPENDIX G
NEW WIDGET GUIDELINES

- Never modify WidgetType or WidgetIdent
- Do not use StyleSet to manage race conditions
- The combining of widget parameters into a compound "XY" run-time parameter is for convenience only. This allows the UA to set two adjacent parameters at once, as no corresponding parameters exist in the Creation Structure.
- Follow existing widgets as much as possible when naming runtime modifiable parameters (e.g., abbreviate longitude as 'Long' not 'Lon')

The following table is provided to assist in completing the new widget's Runtime Modifiable Parameter table.

Table G-9 – Runtime Modifiable Parameter Structure Table Template

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent used in the ParameterStructure

G-10 Widget Library Tables

As part of the inclusion of a new widget in the Section 3.0, Widget Library, the tables identified below will need to be updated.

- Widget Event Cross Reference – Tables 3.1.4-1 and 3.1.4-2
  These tables should be updated.
- Widget Library Summary – Table 3.2.1
  Provide a brief widget description.
- Widget Classification Table – Table 3.2.2-2
  List all applicable categories of the new widget.
- Widget Children – Tables 3.2.3.1
  List all possible parents of the new widget. For a new container widget, list the possible children.
- ARINC 661 Keyword Values – Section 4.6
  Provide input to all applicable tables.
- Possible Widgets for Extension – Section 8.3
  List all possible extensions applied to the new widget.

G-11 Modification Property of Parameters

As described in G-3 – item #8, "All parameters should be definition and run-time modifiable except if this would raise a safety or system issue."

Based on community experience and general best practices, the goal of this section is to help find the balance between the standard approach (everything is both Def-time/Run-time) and the two possible deviations (definition time only parameters and run-time only parameters). In other words, the balance between flexibility and complexity for a given widget.

When possible, examples extracted from the current standard illustrate the described cases.

APPENDIX G
NEW WIDGET GUIDELINES

G-11.1 Cases for Definition Time Only Parameters

Some parameters may be definition time only for the following reasons:

- Static performance analysis (e.g., worst-case) due to limited hardware resources, e.g., memory (maximum strings length, array size, etc.).
- Widget **implementation** simplification.
- Widget behavior simplification, e.g., list of widgets in the broadcast receiver.

G-11.2 Cases for Run-Time Only Parameters

Some parameters may be run-time only for the following reasons:

- When data are not available or useless at definition time, e.g., BufferOfFillStyles in the MapGrid widget.
- Protocol acknowledgement after a CDS initial notification, e.g., EntryValidation in UA Validation.
- Overloading of an existing parameter, e.g., ShiftFirstVisibleEntry in the ScrollList widget, or PosXY in general.

G-11.3 General Cases

These are general rules to apply in all cases:

- Consistency: if several parameters are linked together, they all have the same modification properties for a given widget, e.g., PosX and PosY.

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

APPENDIX H CDS LAYER AND WINDOW MANAGEMENT

H-1 Introduction

This appendix defines a generalized method that can be used by a CDS developer to define and manage all of the layers and windows in a flight deck, using standard ARINC 661 techniques. This is not intended to represent the only method that should be used, but rather, it is intended as a suggestion for how an extension/expansion of traditional ARINC 661 techniques can be used to define a complete display and even entire flight decks. Other methods exist to achieve this capability and they are considered valid approaches.

In this particular method all UA layers for the flight deck are linked together by a single CDS layer, the Super Layer.

Using standard widgets, the Super Layer defines all of the groupings of functional UA layers that are needed to draw each display. The Super Layer uses Connectors to reference those functional UA layers.

A Window Manager UA can then be developed to control the Super Layer and to activate the desired sets of layers in response to inputs from the pilot controls.

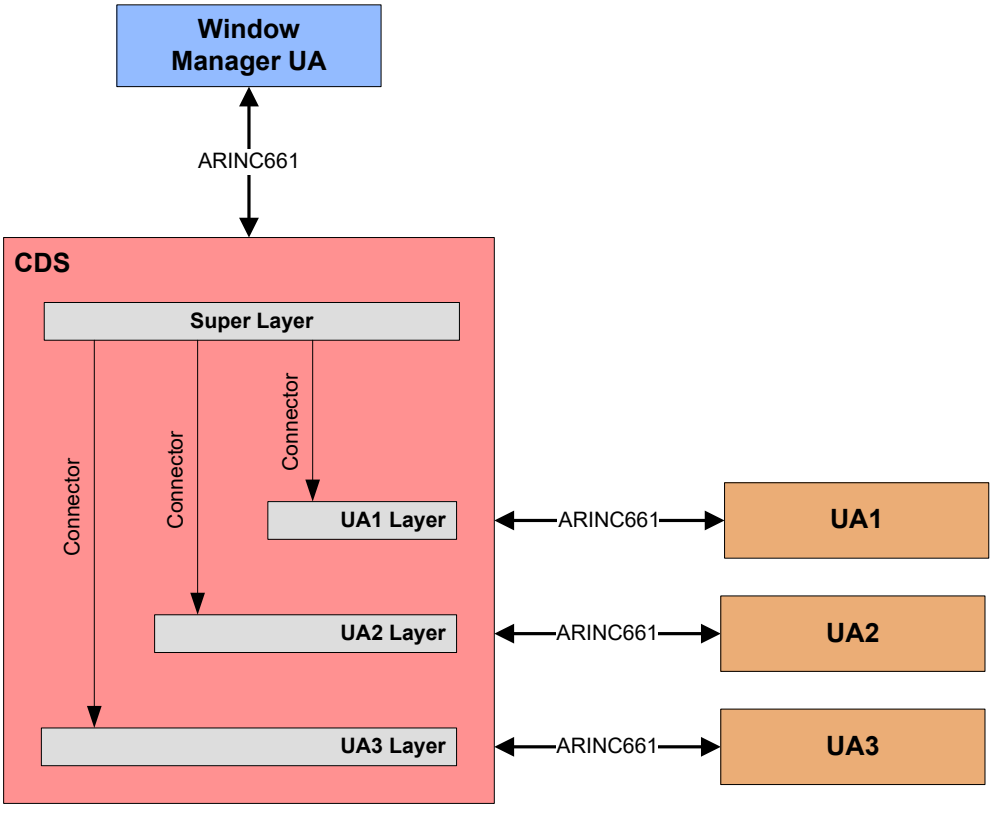


Figure H-1.1 – Super Layer

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

H-2 Super Layer Schema

The hierarchy of the display surfaces controlled by a CDS can be described using the following schema elements. These elements can then be used to create a model of how all of the space on the displays is configured.

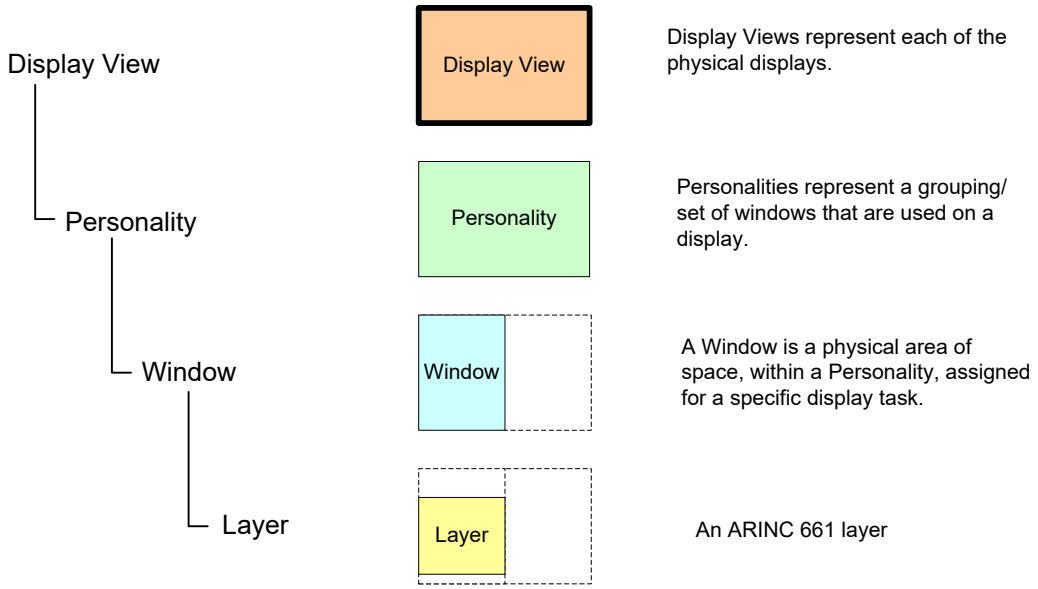


Figure H-2.1 – Display Surface Hierarchy

The schema contains a DisplayView entry for each of the physical displays.

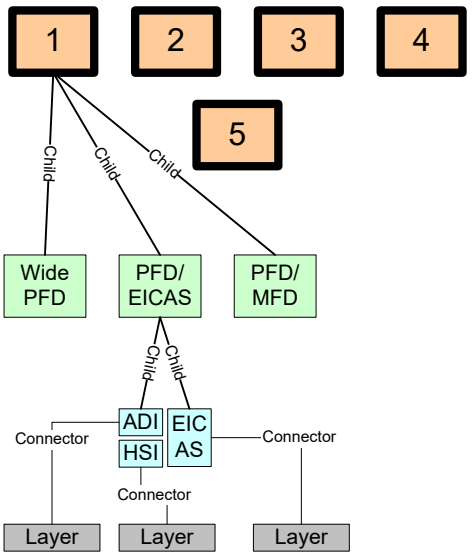
For each DisplayView the schema will have one or more Personalities. A Personality represents a task that the whole display performs (PFD, reverted PFD, MFD, etc.).

For each Personality the schema will have one or more Windows. A Window defines a piece of the screen space that is assigned for a specific display task (e.g., Attitude Director Indicator (ADI), Horizontal Situation Indicator (HSI), Map, Synoptic, etc.).

For each Window the schema will have one or more layers that are necessary to present the associated information. The layers are referenced using Connectors.

An example of this hierarchy of schema elements is shown in Figure H-2.2.

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT



DisplayView 1
- Personality 1.1 (Wide PFD)
  - Window (ADI)
    - Layer (ADI)
  - Window (HSI)
    - Layer (HSI)
- Personality 1.2 (PFD/EICAS)
  - Window (ADI)
    - Layer (ADI)
  - Window (HSI)
    - Layer (HSI)
  - Window (EICAS)
    - Layer (EICAS)
- Personality 1.3 (PFD/MFD)
  - Window (ADI)
    - Layer (ADI)
  - Window (HSI)
    - Layer (HSI)
  - Window (MFD)
    - etc

DisplayView 2
- Personality 2.1 (MFD/EICAS)
- Personality 2.2 (MFD/MFD)
- Personality 2.3 (Wide MFD)
- Personality 2.4 (PFD/EICAS)
- Personality 2.5 (PFD/MFD)

DisplayView 3
- Personality 3.1 (EICAS/MFD)
- Personality 3.2 (MFD/MFD)
- Personality 3.3 (Wide MFD)
- Personality 3.4 (EICAS/PFD)
- Personality 3.5 (MFD/PFD)

DisplayView 4
- Personality 4.1 (Wide PFD)
- Personality 4.2 (/EICAS/PFD)
- Personality 4.3 (MFD/PFD)

DisplayView 5
- Personality 5.1 (MFD/MFD)
- Personality 5.2 (wide MFD)

Figure H-2.2 – Example Schema Hierarchy

H-3 Schema Mapping to Widgets

The elements of this Super Layer scheme can then be mapped to ARINC 661 widgets as shown in Table H-3.

An example of a mapped Super Layer is shown in Figure H-3.

Table H-3 – Schema Mapping to ARINC 661 Widgets

Schema Element	Suggested Widget	Notes
DisplayView	BasicContainer or MutuallyExclusive Container.	The DisplayView is a logical grouping and for most CDS implementations would not require a special custom widget. However, a DisplayView widget could be created to allow widget parameters to be assigned at this level, if required.
Personality	BasicContainer or MutuallyExclusive Container.	The Personality is a logical grouping and for most CDS implementations would not require a special custom widget. However, a Personality widget could be created to allow widget parameters to be assigned at this level, if required.
Window	Panel	The window is a physical demarcation of space on the display. However, for many CDS implementations it also represents a demarcation of processing, memory and other resources. As a result, a custom WindowContainer widget may be required. WindowContainer is described in Section H-6.
Layer	Connector	Wherever the schema refers to a Layer, it does this by reference through a Connector widget. A DataConnector is a special type of connector that allows parameters to be set in its data structure. These parameters are then passed down to the layer when it is activated. The DataConnector is described in Section 3.9.16.

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

MutuallyExclusiveContainer
- MutuallyExclusiveContainer **{Display 1}**
  - BasicContainer **{Personality: Wide PFD}**
    - WindowContainer **{ADI}**
      - Connector **{UA ADI Layer}**
    - WindowContainer **{HSI}**
      - Connector **{UA HSI Layer}**
  - BasicContainer **{Personality: PFD/EICAS}**
    - WindowContainer **{ADI}**
      - Connector **{UA ADI Layer}**
    - WindowContainer **{HSI}**
      - Connector **{UA HSI Layer}**
    - WindowContainer **{EICAS}**
      - Connector **{UA EICAS layer}**
  - BasicContainer **{Personality: PFD/MFD}**
    - WindowContainer **{ADI}**
      - Connector **{UA ADI Layer}**
    - WindowContainer **{HSI}**
      - Connector **{UA HSI Layer}**
    - WindowContainer **{MFD}**
      - MutuallyExclusiveContainer **{Function Selector}**
        - Connector **{Navigation Display}**
        - Connector **{Electronic Checklist}**
        - Connector **{Synoptic}**
        - Connector **{Maintenance}**
        - etc
- MutuallyExclusiveContainer **{Display 2}**
  - BasicContainer **{MFD/EICAS}**
  - BasicContainer **{MFD/MFD}**
  - BasicContainer **{Wide MFD}**
  - BasicContainer **{PFD/EICAS}**
  - BasicContainer **{PFD/MFD}**
- MutuallyExclusiveContainer **{Display 3}**
  - BasicContainer **{EICAS/MFD}**
  - BasicContainer **{MFD/MFD}**
  - BasicContainer **{Wide MFD}**
  - BasicContainer **{EICAS/PFD}**
  - BasicContainer **{MFD/PFD}**
- MutuallyExclusiveContainer **{Display 4}**
  - BasicContainer **{Wide PFD}**
  - BasicContainer **{/EICAS/PFD}**
  - BasicContainer **{MFD/PFD}**
- MutuallyExclusiveContainer {**Display 5}**
  - BasicContainer **{MFD/MFD}**

Figure H-3 – Super Layer Example

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

H-4 Deployment

The deployment of the Super Layer and its corresponding window manager UA is dependent upon the architecture of the CDS. In general, there are two basic types of display system:

- Smart Head
- Remote Head

In its simplest form a smart head is one where the display unit also contains the graphics generation function that receives and processes the ARINC 661 runtime data. The graphics generator only drives one display surface – the one that is co-located with that display unit.

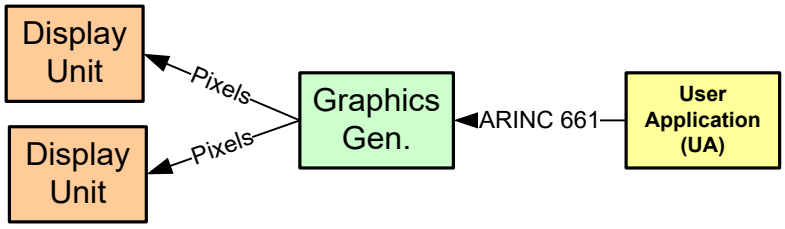


Figure H-4.1 – Remote Head Example

In its simplest form a remote head is one where the graphics generation is performed remotely to the display and “video pixels” are sent to the display unit. A remote graphics generation system may also provide the capability to able to drive any one of the display surfaces in the flight deck.

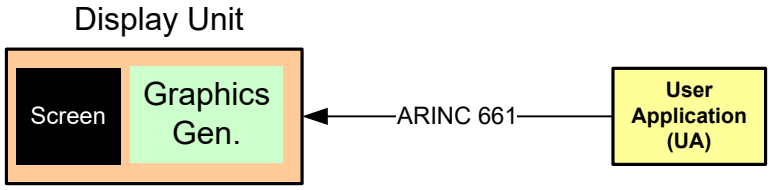


Figure H-4.2 – Smart Head Example

In both architectures the graphics generation function needs to have a Super Layer that defines all possible display functions on all displays.

In a Smart Head system there is typically a desire to have a uniform set of part numbers across the flight deck. In order to do this a display has to have the intelligence to recognize the physical location that it has been placed in and configure itself accordingly. To do this it needs a “complete” Super Layer definition (i.e., a definition of all possible display functions on all displays).

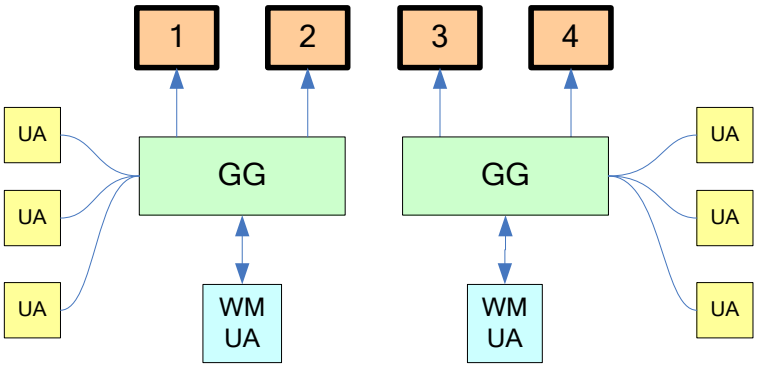
In a Remote Head system, it is likely that the graphic generator will need to be able to drive any display in the event of a display failure. As a result, it also needs to have a “complete” Super Layer definition.

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

H-5 Window Management

The Window Manager UA is the controlling function for the flight deck. Depending upon the system architecture and availability requirements, there will probably be more than one Window Manager UA.

Example, Remote Head System.



Example, Smart Head System.

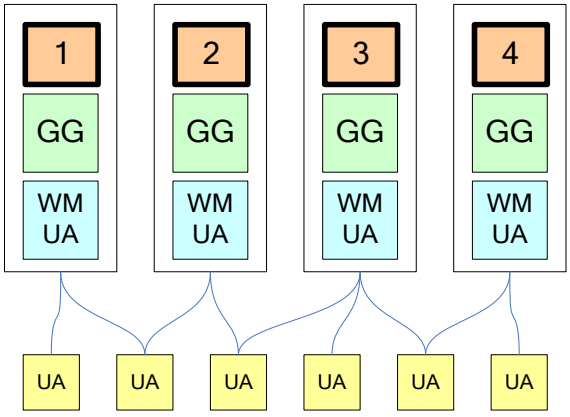


Figure H-5 – Window Management Examples

GG – Graphics Generator. It is the part of the CDS that processes layers and runtime data.

The Window Manager uses inputs from control panels and other input devices to determine what has to be displayed on each of the display units that it controls.

The Window Manager sends runtime parameters to the CDS to set the Visible and VisibleChild states, to select the required DisplayView, Personality and other window control widgets.

The diagrams above show configurations in which Window Manager UAs map one-to-one with GGs. Other architectures are possible. The scheme chosen would affect the way that issues including source selection and failure-handling would have to be addressed. For example, NoServiceMonitor or WatchdogContainer widgets in combination with Connector widgets could be used to help the Window Manager monitor the status of other UAs.

APPENDIX H
CDS LAYER AND WINDOW MANAGEMENT

H-5.1 Window Manager to UA Communication

In many instances the Window Manager may need to pass information down to the layers connected to the Super Layer. This could be used to allow the UA to know certain information about the location of its layer, such as whether it is being drawn on the left or right side of the flight deck. This could also be used to allow the UA to configure different information to be displayed on its layer. Other examples of information passed in this way are listed below:

- Which side of the flight deck
- Which window
- Which display
- Size of the window
- Other window condition information

This Window Manager to UA communication could be achieved using direct, non-ARINC 661 methods; however, there are some advantages of considering a communication path that goes via the Graphics Generator for the following reasons:

- Existing communication paths can be used
- Information can be synchronized with layer activation
- Information can be more easily shared with all UAs

In order to implement this type of Window Manager to UA communication, a widget called DataConnector could be used. The DataConnector is similar to the Connector widget but contains an additional variable length parameter that the Window Manager can use to send a block of data to the UA at layer activation time.

Refer to Section 3.9.16 for information about the DataConnector.

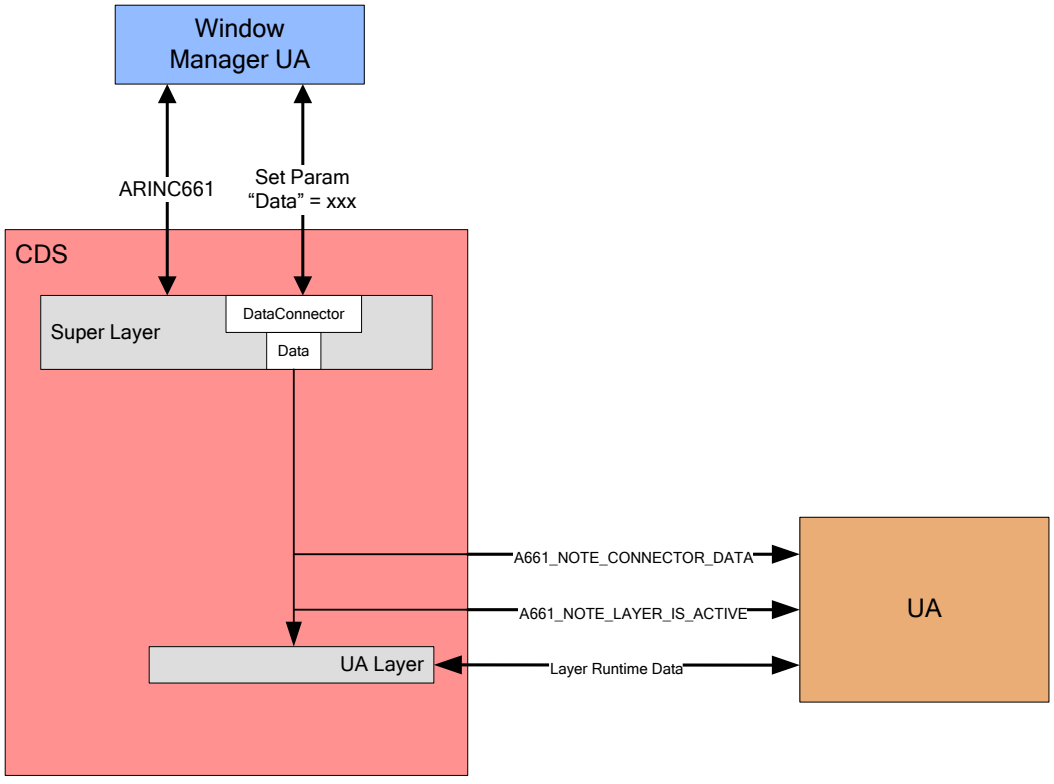


Figure H-5.1 – Window Manager to UA Communication

H-6 Example Custom Widgets

The following section describes custom widgets that may be used by a CDS developer to support the implementation of the Super Layer. These widgets have not been defined in full because the details would be different for each CDS.

H-6.1 WindowContainer

Categories

- Container
- Graphical representation

Description

The WindowContainer is similar to a Panel widget; in addition, the WindowContainer is used to define the overall amount of processing and memory allocated to the layers within it, plus other window related parameters.

Restrictions
Typically only used in the Super Layer.

WindowContainer parameters are defined in Table H-6.1.

Table H-6.1 – WindowContainer widget (example)

Parameter	Change	Description
WidgetType	D	A661_WINDOW_CONTAINER
WidgetIdent	D	Unique identifier of the widget
ParentIdent	D	Identifier of the immediate container of the widget
Visible	DR	Visibility of widget
Enable	DR	Ability of the widget's descendants to be interactive
Pos X	DR	The X position of the widget reference point
Pos Y	DR	The Y position of the widget reference point
Size X	DR	The X dimension (width) of the widget
Size Y	DR	The Y dimension (height) of the widget
MotionAllowed	D	If set to A661_TRUE, it allows the Position and Size of the widget to be modified at runtime
Optional Parameters (examples of other parameters that could be used)		
WindowLocation	D	Indicates the "side" that the window is associated. May be used to define which input devices (i.e., keypads, etc.) are used for interactivity.
ProcessingLimit	D	Defines the maximum time allowed to process the widgets in this window. Effects of exceeding limits are implementation dependent.
Memory Limit	D	Defines the maximum amount of memory that can be used for storing all the widget and other data for this window. Effects of exceeding limits are implementation dependent.
<other parameters>	<?>	Parameters should be added (or deleted) as necessary to define the "functional window parameters" for the target CDS.

H-6.2 DataConnector

The DataConnector is no longer a custom widget since it has been added to the specification (see Section 3.9.16).

APPENDIX I ARINC 661 UA DESIGN CONSIDERATIONS

I-1 Introduction

ARINC 661 defines the communication protocol between User Applications and CDS in Section 4.0, but does not propose a mechanism to facilitate UAs generic access to this protocol.

UA implementers are free to develop any software architecture to abstract the ARINC 661 protocol at any level, or to embed ARINC 661 buffer management logic directly in the UA.

Using a predefined software architecture on the UA side may reduce the overall validation cost for the UAs and simplify the effort of porting a UA from one platform to another.

This appendix identifies possible different types of UA side ARINC 661 Software Architecture; minimum requirements for each architecture-type and guidelines for their design. Other solutions or extensions to these guidelines may be used. Following the scenarios and suggestions described within this guidance appendix is not required to achieve compliance with ARINC 661.

In this appendix, the term “ARINC 661 UA-Software Architecture” will refer to any software architecture which provides UA-side managed (and simplified) access to the communication protocol defined in Section 4 of this document.

I-2 Main Concepts and High-Level View of ARINC 661 UA-Software Architecture

With regard to these guidelines, ARINC 661 contains and defines several things:

- High level concepts, such as windows, widgets and layers
- A detailed protocol standardizing all possible exchanges between a UA and a CDS.

Each CDS implements these concepts. To be able to communicate using these concepts, a UA also needs to implement them.

An implementation of ARINC 661 UA-Software Architecture typically has the following levels of abstraction as a minimum. These result in a logical separation of concerns between architecture layers, providing unit testable parts with clear roles and responsibilities that will be largely unaffected by requirement changes in other architectural layers.

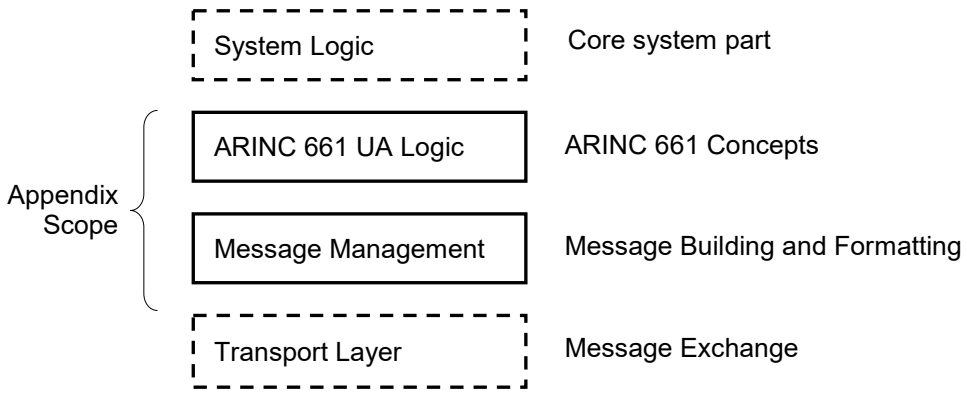


Figure I-2-1 – ARINC 661 Abstractions

- ***Transport Layer*** is responsible for the two-way communication of data between the UA and CDS, it provides those services specific to the choice of message transport (i.e., the bus or network type).
- ***Message Management*** is responsible for the UA/CDS communication and protocol concerns, including message buffer management, message building from sub parts, data encoding, and message decoding.
- ***ARINC 661 UA*** Logic encompasses generic re-usable routines specific to an ARINC 661-enabled UA that enable the system logic to act on high-level concepts (typically Layer and Widget instances) without considering underlying communication issues.
- ***System Logic*** provides features specific to the UA, i.e., equipment logic.

System-Logic and Transport-Layer are outside the scope of this appendix as they are specific to the UA and target operating environment.

I-3 ARINC 661 UA-Software Architecture Design Aims

The following ARINC 661 UA-Software Architecture requirements are common to all UA Software architectures; individual programs may add to this list:

- Reduce the software engineering effort required to build, verify, and certify UAs
- Abstract effort away from messaging concerns, facilitating the focusing of effort on required User-Interface outcomes
- Have common components re-usable by different UAs
- Reduce software complexity and interdependencies between software components – diminishing the impact of requirement changes on the UA software
- Be predictable – Memory Allocation, User Interface experience, and response times must be guaranteed
- Be scalable – to handle many widget instances and layers

I-4 Types of ARINC 661 UA Software Architecture

There are alternate approaches to developing a UA Software Architecture, grouped here into four categories of increasing architecture sophistication.

- Simple Library
- Basic API
- API with Framework
- Full Abstraction (with framework)

I-4.1 Simple Library

The UA calls library routines to assist with the construction and sending of ARINC 661 messages. Incoming ARINC 661 messages are decoded and put onto a queue for action. This simple software architecture introduces common reusable routines, but with divergence from the intent of ARINC Specification 661.

A Simple Library is likely to provide the following components:

- **Buffer Routines** data formatting, storage and retrieval directly related to UA/CDS communication and protocol concerns. They are used to

build/decode a sequence of bytes that can be sent/received from a CDS via the transport layer. May manage message building and likely to include routines to handle data packing, endian, and data-type management.

- **ARINC 661 Helper Routines** directly related to high-level concepts, namely widgets and layers. Hides underlying communication concerns; provide simple means to request layer visibility, set parameter values, etc.

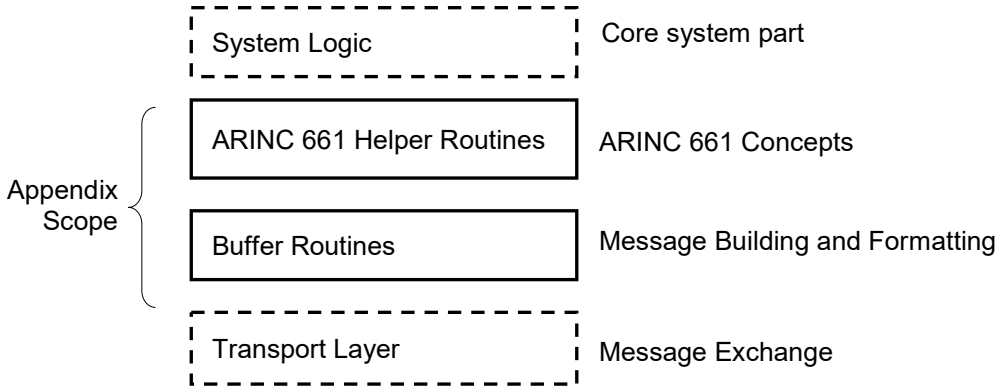


Figure I-4.1-1 – Simple Library UA Software Architecture

I-4.2 Basic API

Provides building blocks for construction of ARINC 661 messages by defining clear routines for manipulating buffers, exposed as an Application Programming Interface (API) at different levels:

Raw Level: At this level buffers are simply seen as bounded arrays of bytes.

- ***Elementary Data Level***: This level provides functions to write or read buffers using application level data types. This notably includes writing and reading of ARINC 661 primitive types such as uchar, ulong, float, BCD, FR, etc. Plus, the management of BIT fields.
- ***Structured Data Level***: Higher level functions can be provided to handle more elaborate structures such as headers, footers, parameters, arrays, blocks, byte alignment, etc.

Each API level can be built on top of previous ones adding more static checks which reduce the risk of errors in buffer manipulation.

The API approach differs from the simple library approach in that it should define the *complete* programming interface and prevent the possibility of extending or circumnavigating the software architecture layer with custom routines.

APPENDIX I
ARINC 661 UA DESIGN CONSIDERATIONS

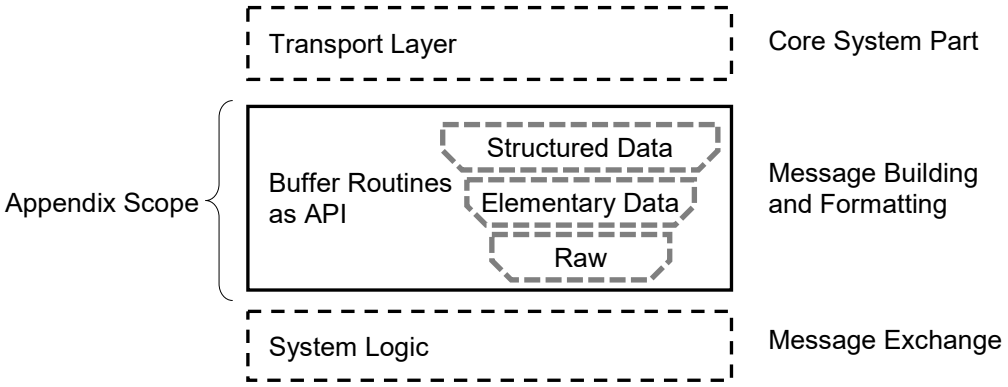


Figure I-4.2-1 – Basic API UA Software Architecture

With a basic API approach the correctness of the buffer structure relative to the ARINC 661 protocol is fully the responsibility of the UA. Validity of parameters relative to CDS configuration should also be handled by the UA.

I-4.3 API with Framework

This approach extends the Basic API by adding an architectural layer to enable the management of ARINC 661 properties without considering buffering routines directly.

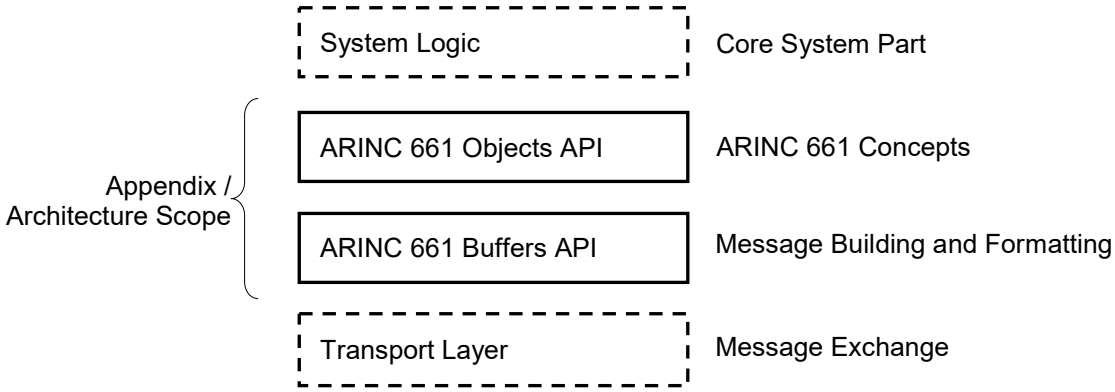


Figure I-4.3-1 – API with Framework UA Software Architecture

I-4.3.1 Components of an API with Framework approach

- ***ARINC 661 Buffers API*** provides data formatting and storage routines directly related to UA/CDS communication and protocol concerns. They are used to define a sequence of bytes that can be sent to a CDS using a channel or used to decode a sequence of bytes that was received from a CDS using a channel.
- ***ARINC 661 Objects API*** are directly related to high-level concepts, namely widgets and layers. These objects are supported by the ARINC 661 protocol. They can be used to hide the underlying communication issues.

It is common to provide representations of the CDS designed widget and layer instances to interact with, these representations (sometimes referred to as proxies) may be auto-generated from the CDS design and hence enforce the validity of parameters and identifiers with the CDS configuration.

I-4.3.2 UA/CDS Interaction Using ARINC 661 Objects

The following figure provides a simplified view of object level interactions between a UA and a CDS during operational phase. Two kinds of objects are considered: one layer proxy and several widget proxies (one for each widget contained in the layer). Each proxy provides all operations necessary to set parameters that are valid with regard to the CDS element that the proxy represents.

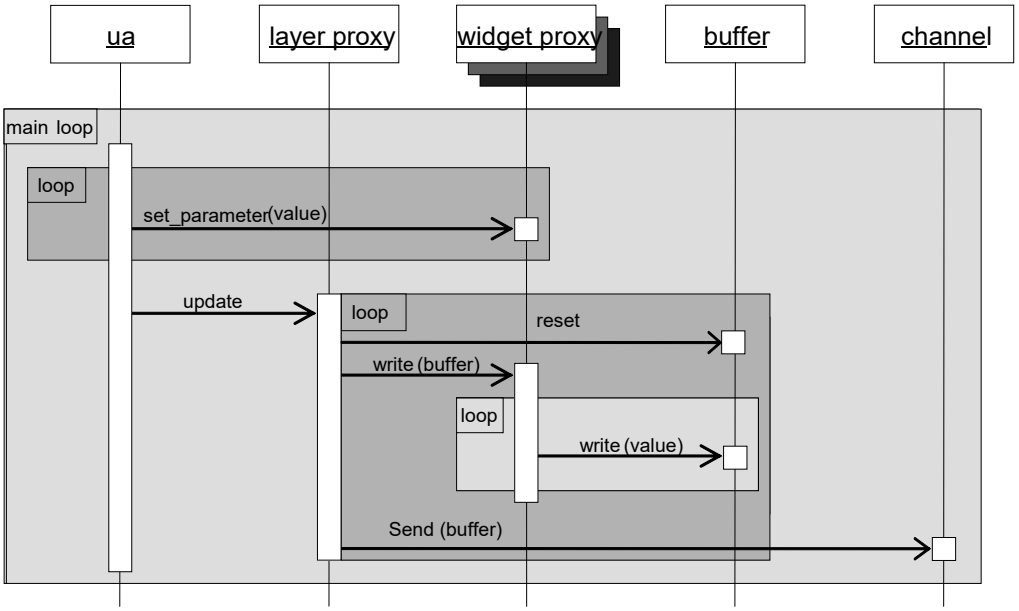


Figure I-4.3.2-1 – UA Production of ARINC 661 Objects

At this level, UA does not directly fill a buffer, but sets parameters of proxy widgets and proxy layers. The UA is unaware of the details surrounding how the buffer is filled. The UA totally delegates filling of the buffer to the proxies.

This example can be extended to handle several layers.

Correctness of the buffer structure is the responsibility of the API. It is the proxy's responsibility to fill buffers in accordance with ARINC 661. Compatibility with CDS configuration must still be handled by the UA.

A similar example can be applied to incoming buffers.

I-4.4 Full Abstraction (with Framework)

This approach differs from the others as it requires the UA to be designed in terms of User-Interface artifacts abstracted above the ARINC 661 widget hierarchy definition.

This approach requires a level of abstraction to be defined, with a supporting set of tools and development process; e.g., a specialized model-based UA development approach coupled with some code generation. A run-time framework is also required that translates back from the abstracted design time User-Interface artifacts to ARINC 661 objects and in turn to ARINC 661 protocol messaging.

This approach would also utilize representations of the CDS designed widget and layer instances to interact with, these representations (sometimes referred to as

proxies) may be auto-generated from the CDS design and hence enforce the validity of parameters and identifiers with the CDS configuration.

I-4.4.1 Recommended Abstraction Approach

Just as the ARINC 661 protocol abstracts “look” from “widget behavior” a similar approach can be introduced UA side. “Behaviors” can be defined that are applied to individual (or collections of) widget instances. Each behavior type defines how its associated widget(s) react to changes in system data and how to inform the system in response to widget events.

Thus a “behavior-type” becomes a new abstracted re-usable component that can be associated with widget instances and system-data.

I-4.4.2 Probable components for this approach:

- ***Buffer Routines*** data formatting, storage, and retrieval directly related to UA/CDS communication and protocol concerns. They are used to build/decode a sequence of bytes that can be sent/received from the CDS via the transport layer. May manage message building and likely to include routines to handle data packing, endian, and data-type management.
- ***ARINC 661 Helper Routines*** directly related to high-level concepts, namely widgets and layers. Hide underlying communication concerns; provide simple means to request layer visibility, set parameter values, etc.
- ***Layer and Widget Update Management*** manages refresh policy, layer activation etc.
- ***Abstraction Definition*** e.g., Behavior type definition and behavior building blocks definition. This is the only layer visible to the UA designer.
- ***Bindings*** are the association of behavior types to widget instances, and behavior parts to system logic. This layer is the UA design definition and differs between UAs (and is not actually part of the architecture).

APPENDIX I
ARINC 661 UA DESIGN CONSIDERATIONS

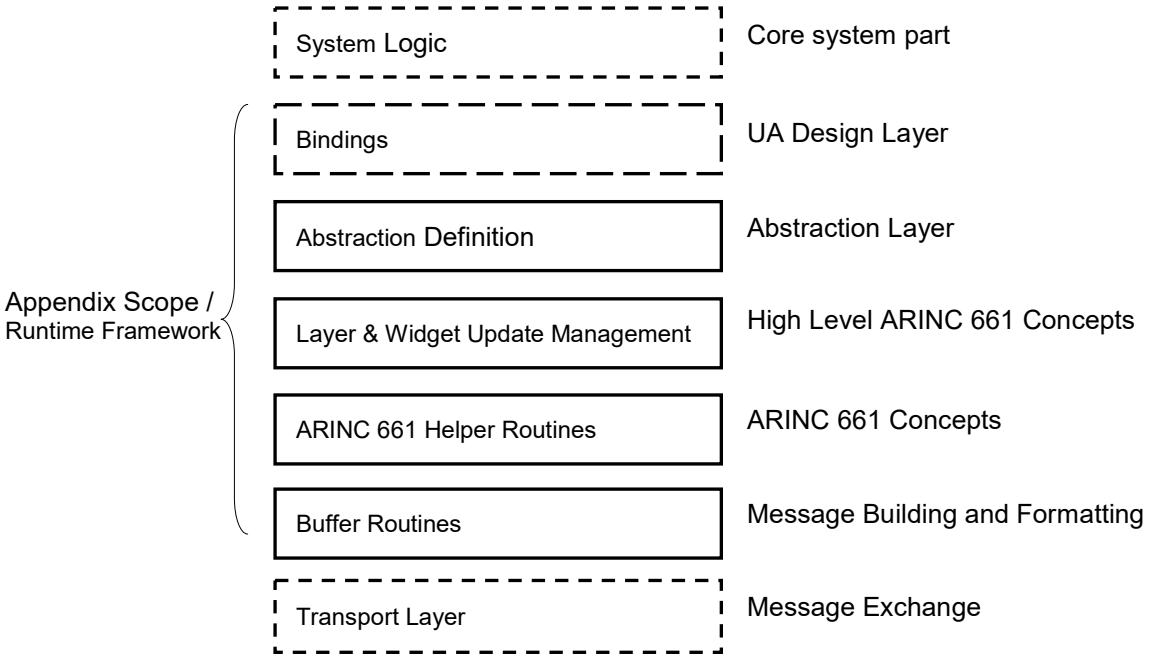
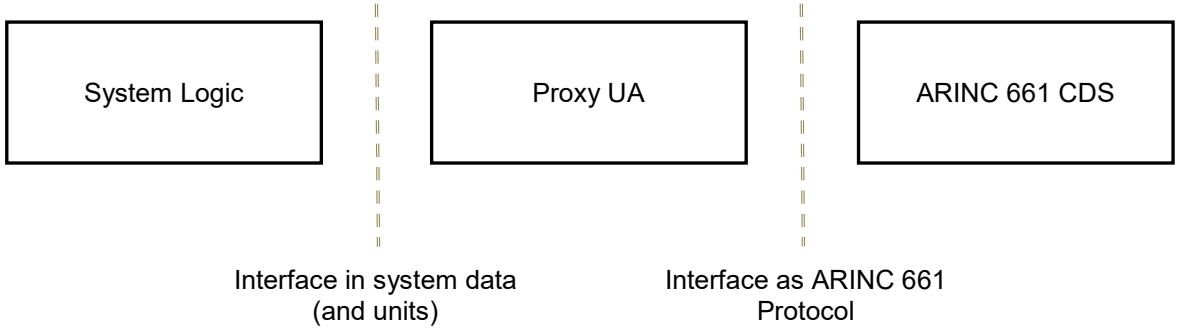


Figure I-4.4.2-1 – Full Abstraction (with Framework)

I-5 Proxy User Applications

The UA architectures with a framework (above) provide an interface to the "core-system part" which enable the use of alternative deployment topologies, including the concept of a Proxy UA.



Where:

System Logic	Provides features specific to the UA, i.e., pure equipment logic without any User Interface concerns.
Proxy UA	Provides the bridge between pure system concerns and User Interface concerns, including a mapping onto the ARINC 661 protocol.
ARINC 661 CDS	Provides the CDS that translates definition files and run-time protocol to provide the certified User Interface experience.

Figure I-5-1 – UA and CDS with a Proxy UA Software Component

The optimal physical deployment of a Proxy UA depends on the overall system requirements, hosting options include:

- **With the core-system equipment (as a separate software component) –** enabling Core-System software re-use on platforms with different UI requirements.
- **As a standalone unit –** enabling legacy equipment to interface with ARINC 661.
- **Within the CDS equipment –** providing the CDS with a “System” interface and minimizing latency effects within the UI.

APPENDIX J LOOK MODELING

J-1 XML Look Specification

As defined in Section 2.2, the term “look” refers to the appearance of widgets, for example, graphical characteristics such as color and border properties. The term “feel” refers to the behavior of widgets, for example, the manner in which an interactive widget reacts to being selected by a crew member. The definition of “look” characteristics is driven by styleset.

The scope of this section is to provide guidelines on how to exchange look information in an ARINC 661 process. In particular:

- Feel is out of scope (however, the look characteristics of each widget are based on the states issued from the feel)
- This appendix does not define any list of required look properties or characteristics. OEM may define and use any look properties they need.

The scope is restricted to the exchange of XML look specification. As a consequence, there is absolutely no assumption regarding the way the widgets look and feel is actually implemented in a CDS.

J-2 Look Capacities/Look Definition

Two elements are necessary regarding look specification:

- The specification of the graphical properties and the states the server can handle for every widget. It is defined in the “look capacities” file
- The definition of the stylesets and their associated graphical properties’ values for each widget. It is defined in the “look definition” file.

Those two elements fulfill different goals. While the Look Capacities is highly dependent on the server capacities and is usually provided by the CDS suppliers (in accordance with HMI needs) at the very beginning of a process with no or very few iterations, the Look Definition can be exchanged on a more frequent basis, with more iterations during the definition phase. This is the reason why two files were designed to exchange those two elements. However, it is possible to exchange the whole specification as a single Look file.

Furthermore, a cross check is possible to verify the consistency of the Look Definition with respect to the Look Capacities. Typically, consistency must be ensured between the properties defined in the Look Capacities and the ones used in the Look Definition.

The following diagram illustrates how those files may intervene in a very high-level process.

APPENDIX J
LOOK MODELING

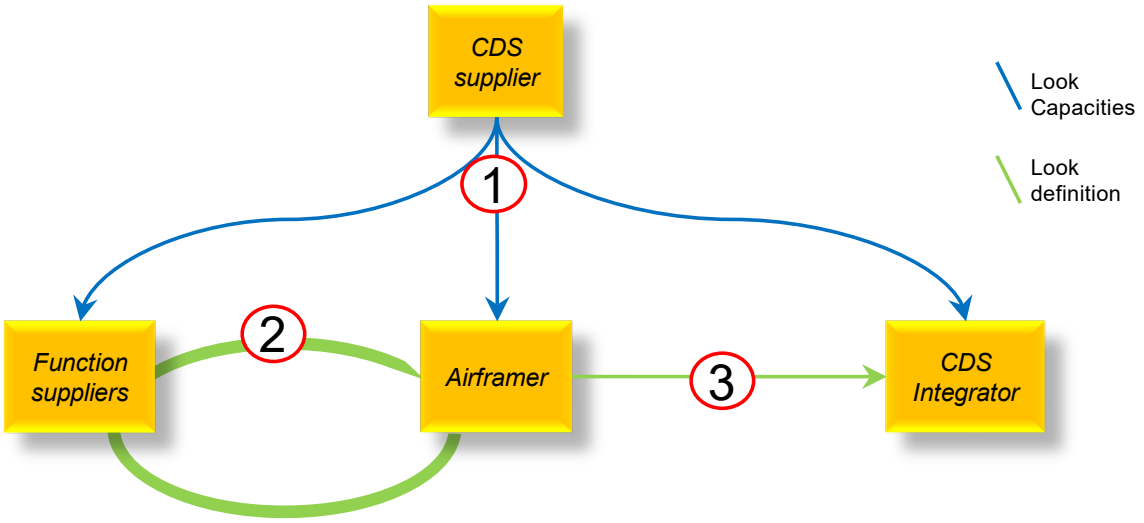


Figure J-2-1 – High Level File Interaction

The following example illustrates with the simple case of a simple push-button the differences between those two files.

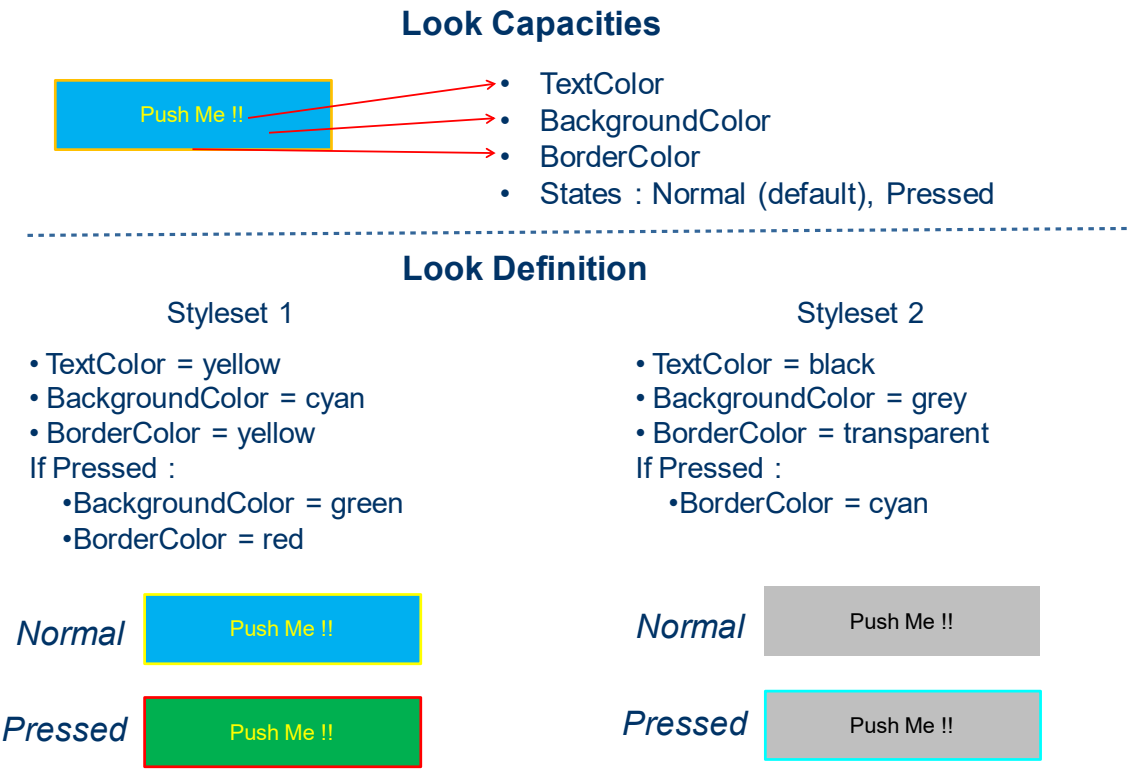


Figure J-2-2 – Look Capacity/Definition File

APPENDIX J
LOOK MODELING

The following list of concepts describes the key drivers of the XML formats provided in Section J-3:

- Globally:
  - Consistency:
    - A “Ref” suffix is used for the attributes referring to the name of an element formally defined (e.g., typeRef attribute’s value must be the name of a type defined in the Look Capacities).
    - Consistency checks can be performed simply using built in XML capacities (see Section J-3).
    - Flexibility: Look Capacities and Look Definition can be exchanged separately or as a single file.
  - Openness:
    - Two categories are defined for the data types. A type can be either basic (e.g., float, long, enumerate) or complex (structured type composed of basic type elements).
    - Every XML element has an optional free text attribute which can be used to provide additional information regarding the element (it can be a simple comment or useful information for a particular OEM).
    - Under every XML element, OEM has the ability to add extensions used by its own implementation. These extensions are identifiable by a dedicated XML name-space (cf. the “W3C Namespaces in XML” 1999 specification) such that it can easily be identified and manipulated. The extension mechanism is described in Section J-4.3.
- Specifically for Look Capacities:
  - Consistency:
    - When an attribute has a restrained range of possible values, this range of values is clearly defined through an enumerate or a constant (e.g., an alignment can only take the values LEFT, RIGHT, or CENTER).
    - When an attribute has a not restrained range of possible values but one or more specific values, the specific values are clearly defined through a constant (e.g., a color that is not restrained but a specific value to define the transparency).
    - The ability to define different values for an attribute depending on the widget’s state is clearly defined through the scope attribute (which can be set to “styleset” or “state”).
  - Avoid redundancy: a set of attributes can be grouped as a “component”. This factorization can be used by different widget look capacities. For example, a COMPONENT_TEXT component can factorize the font size and font color characteristics. This component can then be used by A661_LABEL and A661_PUSH_BUTTON widgets.
  - Openness:
    - The attributes have types (basic or complex), which can be described as needed, along with enumerates and constants used in those types.

- Specifically for Look Definition:
  - Consistency:
    - The Look Definition file refers to the Look Capacities that it must comply with. A cross-check can be performed to ensure that the properties used in the Look Definition are well defined in the referenced Look Capacities.
    - For each widget, all the attributes defined in the Look Capacities must be defined for every styleset.
  - Avoid redundancy:
    - The default values for a widget's attributes are defined for a given styleset. The default values can then be overridden only for states needing different values.
    - Complex data can be defined only through tables (e.g., a table of fonts). Stylesets can reference these complex data.
    - Tables define only complex data.
  - Openness:
    - The set of available values for an attribute can be defined within table elements.

J-3 Consistency Checks Embedded in XML Schemas

For LookCapacities and LookDefinition:

- Constant element name uniqueness whatever the type of the constant.

For LookCapacities:

- Type and enumerate elements name uniqueness,
- Widgetcapacities and componentcapacities elements name uniqueness,
- Literal name uniqueness for each enumerate element,
- Constantproperty name uniqueness for each constant element,
- State name uniqueness for each widgetcapacities element,
- Attribute name uniqueness for each componentcapacities or widgetcapacities element,
- The typeref referenced in a componentcapacities or a widgetcapacities attribute must reference an existing type, componentcapacities or enumerate,
- State names do not contain spaces,
- The id parameter of styleset elements must be an integer between 0 and 65535.

For LookDefinition:

- Typeref attribute of each table element must be unique,
- Componentcapacitiesref attribute of each componentdefinition element must be unique,
- Widgetcapacitiesref attribute of each widgetdefinition element must be unique,
- ID and name uniqueness for entry for each table element,

- Constantproperty uniqueness for each constant element,
- Entryproperty uniqueness for each entry element,
- ID and name uniqueness for styleset element for each componentdefinition or widgetdefinition element
    - Property uniqueness for each componentdefinition styleset or widgetdefinition styleset element, and for each statedefinition in componentdefinition styleset or widgetdefinition styleset.

Cross-consistency:

- Property definition completeness for each componentdefinition and widgetdefinition styleset:
    - For each attribute of scope styleset in the lookcapacities, a corresponding property must be defined in each styleset
    - For each attribute of scope state in the lookcapacities, a corresponding property can be defined in the styleset. If not it should be defined for every state
- This consistency is not checked in the XML Schema. The test must be done by an external tool implemented by the OEM.
- Each componentdefinition element of the lookdefinition must reference a componentcapacities of the lookcapacities,
- Each widgetdefinition of the lookdefinition must reference a widgetcapacities of the lookcapacities,
- The typeref referenced in constant elements of the lookdefinition must reference a type or enumerate of the lookcapacities,
- The typeref referenced in table elements of the lookdefinition must reference a complex type of the lookcapacities,
- The typeref referenced in property elements of the lookdefinition (under componentdefinition or widgetdefinition elements) must reference an attribute of the lookcapacities,
- Name uniqueness for entries present in a table in the lookdefinition and constants with the same type in lookcapacities.

This consistency is not checked in the XML Schema. The test must be done by an external tool implemented by the OEM.

J-4 XML Format for a Look Specification

This section defines the XML format used for ARINC 661 Look Capacities and Look Definition files. It is assumed that the reader is already familiar with XML.

This XML grammar is not specific to any kind of implementation and can be used to describe any server’s look capacities and definitions.

An example is provided in Section J-3.

J-4.1 XML Elements Description

The following section describe the elements that can be used in the Look Capacities and Look Definition files and those which can be used in the Look common file.

APPENDIX J
LOOK MODELING

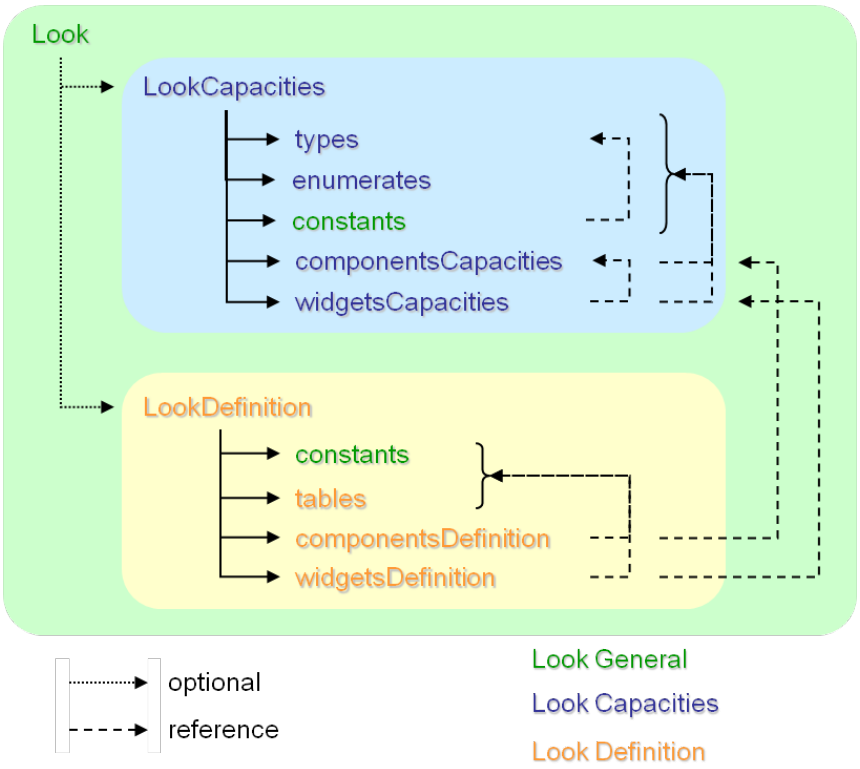


Figure J-4.1-1 – Look Modeling XML Elements

J-4.1.1 Look Capacities and Look Definition XSD (XML Schema Document)

These schemas allow for an automatic check of file compliance with the grammar.

To allow the flexibility of having separate files or a single file for Look Capacities and Look Definition, there is global Look.xsd xml Schema, which is referencing the Look Capacities and Look Definition schemas, which are in turn referencing a LookGeneral.xsd schema defining the global xml elements.

This architecture allows the following organization:

- Look Capacities and Look Definition separate files referencing respectively the LookCapacities.xsd and LookDefinition.xsd XML Schemas.
- Look Capacities and Look Definition separate files referencing the same Look.xsd global XML Schema
- Look Capacities and Look Definition gathered into a single file referencing the Look.xsd global XML Schema
- Separate Look Capacities and Look Definition referenced into a common Look file which is referencing the Look.xsd global XML Schema.

The schemas and associated examples are available as electronic support files within the zip at URL:

https://www.aviation-ia.com/support-files/661-7-look

J-4.1.2 Naming Rules Recommendations

For the values of all xml attributes “name”, it is recommended to define a specific naming rule to avoid ambiguity and to ease code generation.

For instance, the value of the attribute could be composed by only:

- all characters between “a” and “z” or all characters between “A” and “Z”
- digits (“0” to “9”) but not for the first character
- the character underscore “_”

In this case, “LOOK_ARW” , “Blue1” , “_Type2OfColor” will be authorized, but “1LOOK_ARW” , “Blue 1” , “/type_of_color” will not be.

J-4.1.3 Reserved Words XML Elements Description

The special value “NOT_RELEVANT” for “value” xml attribute must be only used when the value is not relevant.

Consequently:

- “NOT_RELEVANT” string cannot be used for the value of all “name” attributes.

An example of use of the “NOT_RELEVANT” special value could be:
A bitmap type is defined in which two properties are defined:

- format
- transparent color

According to the bitmap format, the transparent color property could have no sense.

J-4.1.4 Optional Default Values for the Attributes in the Look Capacities

It is possible to specify the default value of an **attribute** element. It is especially useful to do this for **attribute** elements which are defined through the StyleSet extension array (because some widgets might have attributes defined in a **styleExtClass** element, but might not have the StyleSet extension defined in their Definition File). It is performed by using the “**value**” or “**valueRef**” sub-element. Note that as it is defined in the Look Capacities, the “**value**” or “**valueRef**” sub-elements must refer to constants defined in the Look Capacities.

```
<attributes>
   <attribute name="BorderSize" typeRef="int" scope="styleset">
      <value>10</value>
   </attribute>
   <attribute name="BorderColor" typeRef="color" scope="state">
      <valueRef>DEFAULT_COLOR</valueRef>
   </attribute>
   <attribute name="BackgroundBitmap" typeRef="bitmap" scope="styleset">
      <valueRef>NO_BITMAP</valueRef>
   </attribute>
</attributes>
```

J-4.1.5 Optional Bitmask Position and Size for the Attributes in the Look Capacities

It is possible to specify the bit mask where each **attribute** element may be found within the StyleSet ushort value. In this case, the two optional **bitPos** and **bitSize** properties define the position and size of the bit mask for the **attribute** element. In the following example, the “StyleSetText” **attribute** element is defined from the bit 2 to bit 4 of the StyleSet ushort value (bit 0 is the first bit).

```
<attributes>
  <attribute name="BorderSize" typeRef="int" scope="styleset"
             bitPos="0" bitSize="2" />
  <attribute name="StyleSetText" typeRef="C_Text" scope="state"
             bitPos ="2" bitSize ="2" />
  <attribute name="BorderColor" typeRef="color" scope="state"
             bitPos ="4" bitSize ="4" />
  <attribute name="BackgroundBitmap" typeRef="bitmap" scope="styleset"
             bitPos ="8" bitSize ="8" />
</attributes>
```

Note that because the **bitPos** and **bitSize** XML attribute tags define the position and size of a bit mask within an ushort, bitPos + bitSize must not be greater than 16.

Note that bitmask positions and sizes must be defined for all the **attribute** elements or not defined for any **attribute** element in the Look Capacities:

- If bitmask positions and sizes are specified, then the Look Definition file should not be used, because in that case the specification about how to get the values for the **attribute** elements is completely specified in the Look Capacities.

J-4.1.6 Relationship with the StyleSet Extension

Additional StyleSet extension StyleSets are defined through the **styleExtClass** element (in the Look Capacities) and the **styleExtClassDef** (in the Look Definition):In the Look Capacities: each styleExtClass element specifies the attributes for one index of the StyleSet extension array. Each **attribute** element is defined globally once for the widget:

- If the **attribute** element is not present in any **styleExtClass** element, it is only defined through the non-extended part of the Look. It must have the XML attribute tag “**definitionType**” set to **“styleset,”** or XML attribute tag should not be present for this **attribute** element
- If the **attribute** element is present in one **styleExtClass** element, it is defined through the extended part of the Look. It must have the XML attribute tag “**definitionType**” set to **“extension”**

The way the StyleSet extension indexes in the StyleSet array are defined impose the following rules:

- Every **attrRef** present in the **styleExtClass** must refer to a corresponding **attribute** element declaration for the widget. However, as it is just a reference, the type of the corresponding **attribute** element is defined in the **typeRef** XML attribute tag declaration for this element
- It is not allowed to have the same **attrRef** present in more than one **styleExtClass**
- The **attribute** elements which are in the **attributes** declaration but not in any **styleExtClass** are those which are managed by the “not extended” part of the StyleSet

APPENDIX J
LOOK MODELING

Example:

In the following example:

- The stylesettext **attribute** is defined through the non-extended styleset part of the widget
- The bordersize and bordercolor **attributes** are defined through the first index of the styleset extension array
- The backgroundbitmap **attribute** is defined through the second index of the styleset extension array

```
<attributes>
   <attribute name="BorderSize" typeRef="int" scope="styleset"
              definitiontype="extension"/>
   <attribute name="StyleSetText" typeRef="C_Text" scope="state"
              definitiontype="styleset"/>
   <attribute name="BorderColor" typeRef="color" scope="state"
              definitiontype="extension" />
   <attribute name="BackgroundBitmap" typeRef="bitmap" scope="styleset"
              definitiontype="extension" />
</attributes>
<styleExtClasses>
   <styleExtClass index="0" name="borderStyle">
      <attrRef nameRef="BorderSize"/>
      <attrRef nameRef="BorderColor" />
   </styleExtClass>
   <styleExtClass index="1" name="background">
      <attrRef nameRef="BackgroundBitmap"/>
   </styleExtClass>
</styleExtClasses>
```

As for the non-extended part of the StyleSet, it is possible to define the position and size of the **attribute** bit mask in the StyleSet extension array for its associated index through the **bitPos** and **bitSize** XML attribute tags.

Example:

```
<attributes>
    <attribute name="BorderSize" typeRef="int" scope="styleset" definitiontype="extension" />
    <attribute name="StyleSetText" typeRef="C_Text" scope="state" bitPos="8" bitSize="8"
definitiontype="styleset"/>
    <attribute name="BorderColor" typeRef="color" scope="state" definitiontype="extension" />
    <attribute name="BackgroundBitmap" typeRef="bitmap" scope="styleset"
definitiontype="extension" />
 </attributes>
 <styleExtClasses>
    <styleExtClass index="0" name="borderStyle">
       <attrRef nameRef="BorderSize" bitPos="0" bitSize="8"/>
       <attrRef nameRef="BorderColor" bitPos="8" bitSize="8"/>
    </styleExtClass>
       <styleExtClass index="1" name="background">
           <attrRef nameRef="BackgroundBitmap" bitPos="0" bitSize="16"/>
        </styleExtClass>
      </styleExtClasses>
```

In the above example:

- The StyleSetText is defined at the **attribute** element level (not in the extension). Therefore, the position and size of this **attribute** element are bit-masks in the StyleSet 16-bit parameter. Note that the presence of a **bitPos** and **bitSize** for an **attribute** element is another way to define whether an **attribute** is defined through the StyleSet or through an extension

- Within the array element of index 0 of the 16-bit array element (A661_STYLE_SET_ARRAY), the first 8 bits are used for the BorderSize, and the next 8 bits for the BorderColor
- Within the array element of index 1, all the 16 bits are used for the BackgroundBitmap

J-4.2 Look Capacities and Look Definition XSD (XML Schema Document)

This schema defines the interface syntax, superseding the DTD equivalent, which is now deprecated. The schema is available as an electronic support file within the zip at URL:

https://www.aviation-ia.com/support-files/661-7-look

J-4.3 The Schemas and Associated Examples are Available Under the Same URL.Extension Mechanism

The OEM has the ability to extend the Look format for its own specific needs. The OEM must define each extension in a separated XML name-space.

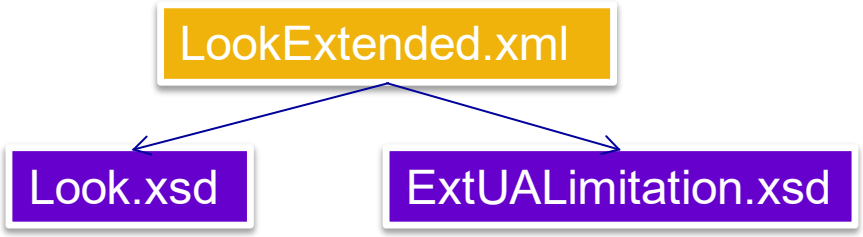


Figure J-4.3-1 – Look Extension Mechanism

For instance, an objective of an extension in the Look format could be to indicate that a specific styleset can only be used with a limited number of UAs.

The following XML Schema Document (ExtUALimitation.xsd) presents an example of how to define this extension in a separated name-space:

```
<xs:schema
    xmlns="http://www.acompany.com/ExtUALimitation"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.acompany.com/ExtUALimitation"
    elementFormDefault="unqualified">
       <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
       <xs:annotation>
              <xs:documentation>This is an example of a possible way to define an extension
in a separated XML namespace
              </xs:documentation>
       </xs:annotation>
       <xs:complexType name="commentType" mixed="true"/>
       <xs:complexType name="UALimitationListType">
              <xs:sequence>
                     <xs:element ref="comment" minOccurs="0"/>
                     <xs:element ref="UARef" maxOccurs="unbounded"/>
              </xs:sequence>
       </xs:complexType>
       <xs:complexType name="UARefType">
              <xs:sequence>
                     <xs:element ref="comment" minOccurs="0"/>
              </xs:sequence>
              <xs:attribute name="name" type="xs:string" use="required"/>
       </xs:complexType>
       <xs:element name="UALimitationList" type="UALimitationListType"/>
```

```
        <xs:element name="UARef" type="UARefType"/>
        <xs:element name="comment" type="commentType"/>
</xs:schema>
```

And this is an example of how to use this extension in the Look Definition XML file (LookExtended.xml):

```
<LookDefinition
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.aviation-ia.com/support-files/661-7-look/2"
    xmlns:ext="http://www.acompany.com/ExtUALimitation"
    xs:schemaLocation="http://www.aviation-ia.com/support-files/661-7-look/2 Look.xsd
http://www.acompany.com/ExtUALimitation ExtUALimitation.xsd">
...
  <widgetsDefinitions>
    <widgetDefinition widgetCapacitiesRef="A661_PUSH_BUTTON">
    ...
      <styleset id="10" name="Style10">
        <ext:UALimitationList>
          <ext:comment>this is an idea on how to use the concept of extensions for specify
                      this styleset can only be used with these UAs</ext:comment>
          <ext:UARef name="FMS" />
          <ext:UARef name="TCAS" />
        </ext:UALimitationList>
        <properties>
          <property attributeRef="BorderSize">
            <valueRef>LARGE_SIZE</valueRef>
          </property>
        </properties>
        <stateDefinition>
          <statesList>
            <stateItem stateRef="FOCUS"/>
          </statesList>
          <properties>
            <property attributeRef="BorderColor">
              <valueRef>BLUE</valueRef>
            </property>
          </properties>
        </stateDefinition>
      </styleset>
    </widgetDefinition>
    ...
  </widgetsDefinitions>
</LookDefinition>
```

J-4.4 Look Capacities and Look Definition XML examples

A simple example with a single widget (A661_PUSH_BUTTON) of the xml files defined for the Look Capacities and Look Definition (defined in a single Look.xml file) is available at the same URL as the Schema.

APPENDIX K UA/CDS INTERFACE

K-1 UA/CDS Interface Specification

A UA/CDS interface is a non ambiguous specification of the exchanges between a UA and a CDS, for a given DF. Figure K-1-1 illustrates where this UA interface is located in a typical high level architecture.

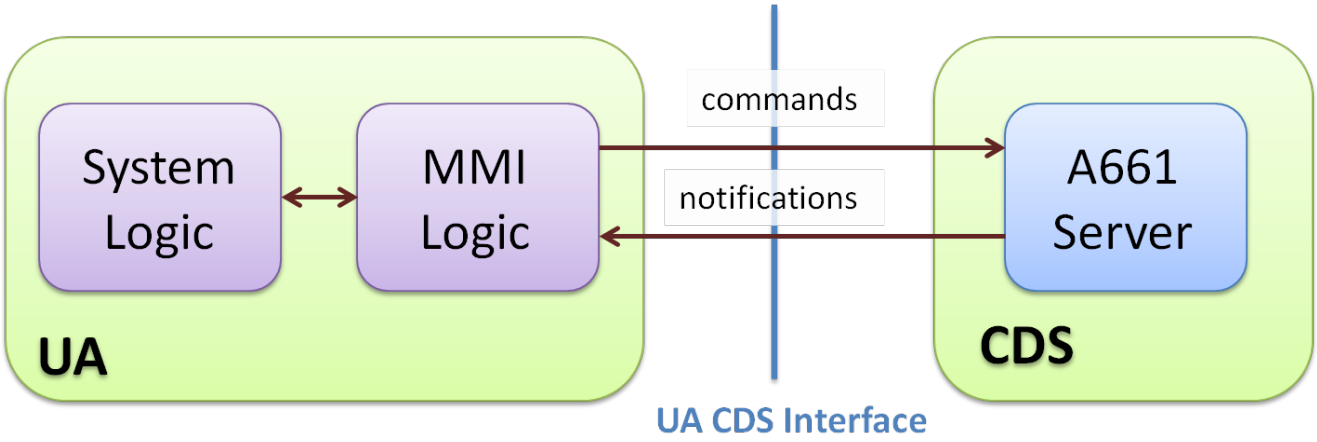


Figure K-1-1 – UA/CDS Interface

The intent of such UA/CDS interface is to specify the flow of the commands and notifications that are actually used by the UA in runtime among all of those offered by the DF.

For example, in the simple DF illustrated in Figure K-1-2, we have a dozen widgets representing roughly 100 possible commands. However, less than 10 commands are enough to achieve the desired UA behavior.

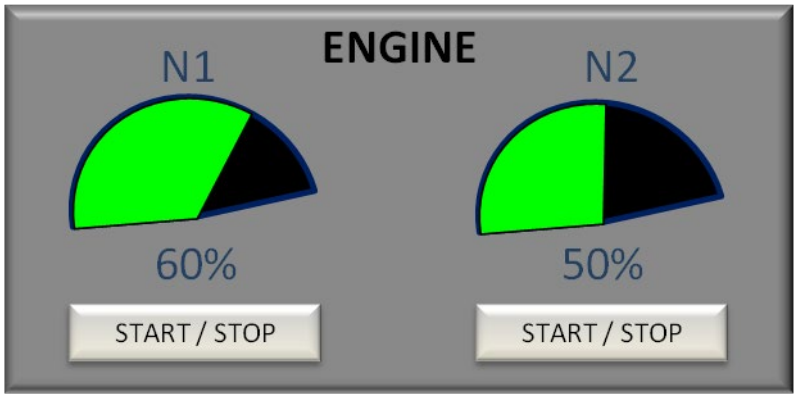


Figure K-1-2 – Example Graphics

Moreover, some usage constraints for this example are:

- The readout and arc circle end angle values are piloted with the same functional value,
- The readout and arc circle end angle values must be refreshed simultaneously.

Apart from the non ambiguity of this UA/CDS interface, facilitating the exchange of specification, additional benefits can be a better support of UA/CDS design and

integration activities such as bandwidth characterization, worst case analysis and use case elaboration for system incremental Verification and Validation.

The UA/CDS Interface definition can be increased in scope to cover other concepts (buffer structure, MMI logic, etc.) using the provided extension mechanism.

K-2 UA/CDS Interface Model Overview

The following list describes the main concepts supported by the model:

- Filtering: only commands and notifications actually used by the UA are defined in the UA/CDS Interface.

  Note: By commands, we mean A661 setParameters and Requests as defined in the standard, notifications referring to A661 events and errors.

- Logical abstraction: the UA only uses the notion of command interfaces and notification interfaces instead of using directly A661 commands and notifications allowing more flexibility on their usage.
- Factorizing: commands (or notifications) related to several widgets can be considered as one single command interface (or notification interface) by the UA (Figure K-2-1).
- Interfaces grouping: command interfaces and notification interfaces can be grouped to reflect their logical link (Figure K-2-1).
- Graphical grouping: command and notification can be grouped to reflect their functional link (Figure K-2-1). For instance, the commands associated to a given functional group could always be sent into the same A661 buffer to ensure graphical layer refresh consistency.
- Sending modalities: commands can be sent according to different rules (e.g., the command can be sent cyclically or upon modification only).
- Sending priorities: a priority can be associated to a command to indicate most significant commands. For factorizing purpose, the command priority can be inherited from parent list of commands or graphical group. For instance, this information can be used to fill the A661 buffer from commands with highest priority to commands with lowest ones.
- Separation between:
  - The description of the UA Message Management capacities (such as on what conditions the updates from the UA are sent to the CDS, for instance cyclically, or when the value changes).
  - And the interface definition.

APPENDIX K
UA/CDS INTERFACE

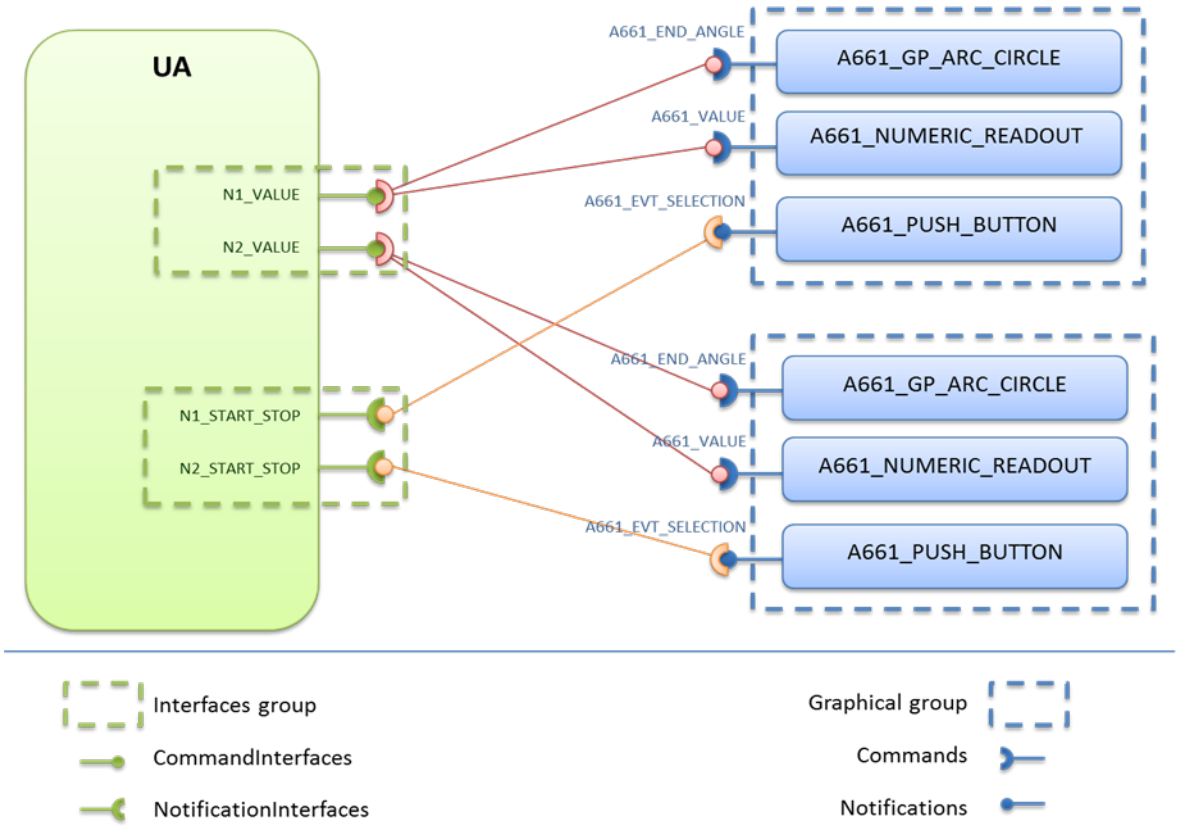


Figure K-2-1 – Links CommandInterface-Command and NotificationInterface-Notification

K-2.1 UA CDS Interface Capacities/UA CDS Interface Definition

Two elements are necessary regarding UA CDS Interface specification:

- UA CDS Interface Capacities: the specification of the UA Message Management capacities (such as on what conditions the updates from the UA are sent to the CDS, for instance cyclically, or when the value changes). It is defined in the “UA CDS Interface Capacities” file.
- UA CDS Interface Definition: the specification of the flow of the commands and notifications that are actually used by the UA in runtime among all of those offered by the DF.

These two elements fulfill different goals. While the UA CDS Interface Capacities specify the available message management schemes for one or several UAs, and usually does not change frequently, the UA CDS Interface Definition can be exchanged on a more frequent basis, with more iterations during the definition phase. This is the reason why two files were designed to exchange those two elements. However, it is possible to exchange the whole specification as a single UA CDS Interface file.

A cross check is possible to verify the consistency of the UA CDS Interface Definition with respect to the UA CDS Interface Capacities. Typically, consistency must be ensured between the properties defined in the UA CDS Interface Capacities and the ones used in the UA CDS Interface Definition.

APPENDIX K
UA/CDS INTERFACE

K-3 XML Model Key Drivers

The following list of concepts describes the key drivers of the xml formats provided in Section K-5:

- Consistency:
  - A “Ref” suffix is used for the attributes referring to the name of an element formally defined.
  - Consistency checks can be performed simply using standard XML library routines (see Section K-4).
- Openness:
  - Every XML element has an optional free text attribute which can be used to provide additional information regarding the element (it can be a simple comment or useful information for a particular OEM).
  - Under every XML element, OEM has the ability to add extensions used by its own implementation. These extensions are identifiable by a dedicated XML name-space (cf. the “W3C Namespaces in XML” 1999 specification) such that it can easily be identified and manipulated. The extension mechanism is described in Section K-5.3
- Flexibility
  - The format is designed to support the description of any UA/CDS interface, without ambiguity nor restriction.
  - UA/CDS Interface Capacities and Definition can be exchanged separately or as a single file.

K-4 Consistency Checks Embedded In XML Schemas

- commandInterface element name uniqueness
- notificationInterface element name uniqueness
- setParameter element (layerIdent, widgetIdent, widgetParameter) uniqueness
- request element (layerIdent, layerRequest) uniqueness
- layerNotification element (layerIdent, layerNotification) uniqueness
- widgetEvent element (layerIdent, widgetIdent, widgetEvent) uniqueness
- widgetError element (layerIdent, widgetIdent, widgetError) uniqueness
- layerError element (layerIdent, layerError) uniqueness
- for a given notificationData element, notificationAttributeName is uniquely defined
- graphicalGroup element name uniqueness
- interfacesGroup element name uniqueness

K-5 XML Format for a UA/CDS Interface

This section defines the XML format used for ARINC 661 UA/CDS Interface files. It is assumed that the reader is already familiar with XML.

This XML grammar is not specific to any kind of implementation and can be used to describe any UA/CDS interface.

APPENDIX K
UA/CDS INTERFACE

A detailed example is provided in Section K-5.4.

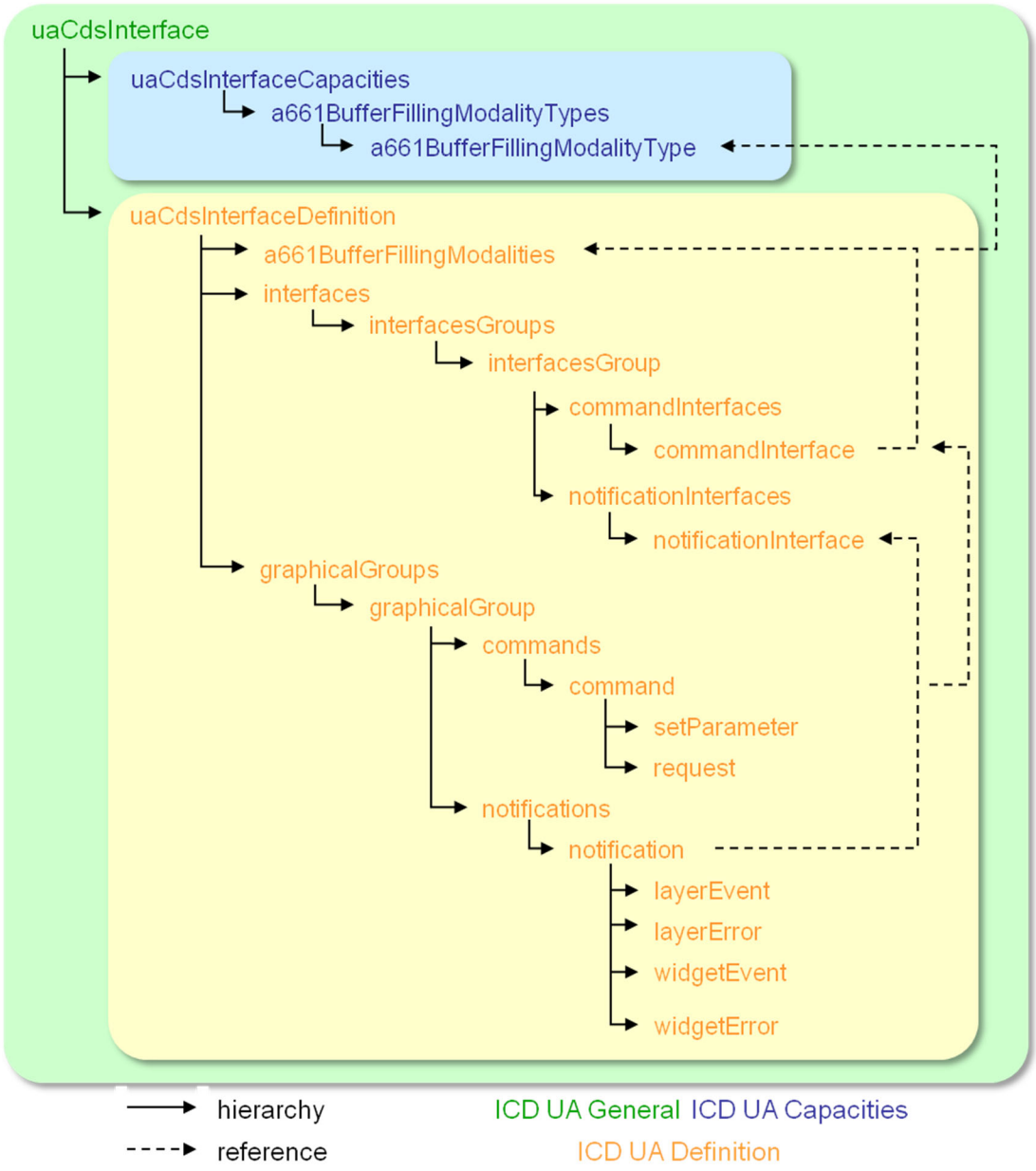


Figure K-5-1 – UA/CDS Interface XML Elements

K-5.1 UA/CDS Interface XSD (XML Schema Document)

These schemas define the interface syntax. In order to allow the flexibility of having separate files or a single file for UA/CDS Interface Capacities and Definition, there is a global UACDSInterface.xsd xml Schema, which is referencing the UACDSInterfaceCapacities and UACDSInterfaceDefinition schemas, which are in turn referencing an UACDSInterfaceGeneral.xsd schema defining the global xml elements.

The schemas and associated examples are available as electronic support files within the zip at URL:

https://www.aviation-ia.com/support-files/661-7-uacdsinterface

APPENDIX K
UA/CDS INTERFACE

K-5.2 Extension Mechanism

The OEM has the ability to extend the UA CDS Interface format for its own specific needs. The OEM must define each extension in a separated XML name-space.

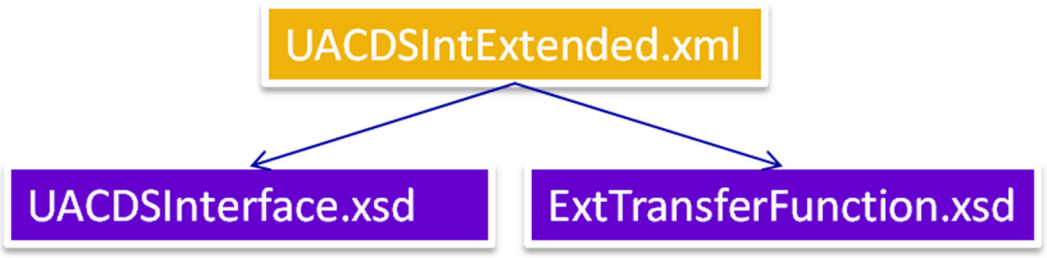


Figure K-5.2-1 – UA/CDS Extension Mechanism

For instance, an objective of an extension could be to associate a logical transfer function to a command interface.

The following XML Schema Document (ExtTransferFunction.xsd) presents an example of how to define this extension in a separated name-space:

```
<xs:schema
    xmlns=http://www.acompany.com/UACDSInterfaceExtension
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.acompany.com/UACDSInterfaceExtension"
    elementFormDefault="unqualified">
        <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
        <xs:annotation>
                <xs:documentation>This is an example of a possible way to define UA CDS Interface
logical transfer function

              </xs:documentation>

        </xs:annotation>
        <xs:complexType name="transferFunctionExtType">
                <xs:sequence>
                        <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="variable" maxOccurs="unbounded"/>
                        <xs:element ref="transferFunction"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="variableType">
                <xs:attribute name="name" type="xs:string" use="required"/>
        </xs:complexType>
        <xs:complexType name="transferFunctionType" mixed="true">
                <xs:attribute name="value" type="xs:string"/>

        </xs:complexType>

        <xs:element name="transferFunctionExt" type="transferFunctionExtType"/>

        <xs:element name="transferFunction" type="transferFunctionType"/>

        <xs:element name="variable" type="variableType"/>

</xs:schema>
```

And this is an example of how to use this extension in the UACDS Interface XML file (UACDSInterface.xml):

```
<uaCdsInterface applicationId="1" definitionFileRef="TEST.df"
                xmlns="http://www.aviation-ia.com/support-files/661-7-uacdsinterface/1"
                xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ext="http://www.acompany.com/UACDSInterfaceExtension"
                xs:schemaLocation="http://www.aviation-ia.com/support-files/661-7-uacdsinterface/1
http://www.acompany.com/UACDSInterfaceExtension ExtTransferFunction.xsd">
   <uaCdsInterfaceCapacities>
   ...
   </uaCdsInterfaceCapacities>
   <uaCdsInterfaceDefinition>
      <a661BufferFillingModalities defaultA661BufferFillingModality="Changing">
      ...
      </a661BufferFillingModalities>
      <interfaces>
         <commandInterfaces>
            <commandInterface name="LAYER_VISIBILITY">
                <ext:transferFunctionExt>
                    <freeText>this is an idea on how to use the concept of extensions for specify that
this commandInterface is associated with a transfer function : LAYER_VISIBILITY = FMSActive and
ILSActive
                    </freeText>
                    <ext:variable name="FMSActive"/>
                    <ext:variable name="ILSActive"/>
                    <ext:transferFunction value="FMSActive and ILSActive">
                    </ext:transferFunction>
                </ext:transferFunctionExt>
            </commandInterface>
            ...
         </commandInterfaces>
      ...
      </interfaces>
   ...
   </uaCdsInterfaceDefinition>
</uaCdsInterface>
```

K-5.3 UA/CDS Interface XML Example

A detailed example XML file defined for the UA interface is available within the same zip as the Schema at URL:

https://www.aviation-ia.com/support-files/661-7-uacdsinterface

APPENDIX L ASCII EXTENDED CHARACTER SET MAPPINGS

L-1 Introduction

This section defines an XML format that specifies how ASCII Extended characters map to UNICODE code points. Refer to Section 3.2.5.1 for an overview of DF character encodings. Refer to Section 6.2.1.1 for how ASCII Extended character set mappings are referenced by XML DF files.

L-2 ASCII Extended Character Set Mapping XSD (XML Schema Document)

The schemas and associated examples are available as electronic support files within the zip at URL:

https://www.aviation-ia.com/support-files/661-7-charmap

L-3 Additional Commentary

XML Representation:

- When xmlrepr =backslash_escape in a mapcharblock or mapchar XML element, character(s) are to be backslash escaped (i.e., use \xHH format) in the XML DF.
- When xmlrepr=direct or xmlrepr is omitted on a mapcharblock or mapchar XML element:
  - Special characters (‘&’, ‘<’, ‘>’, single quote or double quote), use the appropriate special predefined XML entity: “&”, “<”, “>”, “'” or “"” respectively.
  - Other (non-special) characters are directly encoded into the XML DF using UTF-8.
- Characters that are not declared in a mapchar or mapcharblock XML element (e.g., custom characters that do not map to a UNICODE code point) are to be backslash escaped in the XML DF.
- It is recommended that all control characters be backslash-escaped in the XML DF.

Handling of the following error cases is implementation-dependent:

- The same character being specified more than once in an ASCII Extended character set mapping.
- Invalid XML characters (see Section 6.2.3) or the newline character (0x0A) being specified in an ASCII Extended mapping file with xmlrepr=direct or xmlrepr omitted.

ASCII Extended character mapping files can serve a specification purpose even if they are not explicitly supported as part of a tool chain.

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

SUPPLEMENT 1
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACE TO USER SYSTEMS

Published: June 26, 2003

Prepared by the Airlines Electronic Engineering Committee

Adopted by the Airlines Electronic Engineering Committee: March 6, 2003

A. PURPOSE OF THIS DOCUMENT

This supplement introduces various changes and additions to ARINC Specification 661. It adds eight new widgets to the standard and provides clarification of the definition of the Cockpit Display System (CDS) interface to user systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document, vertical change bars in the margin will indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

0.0 Global Changes to Nomenclature Used in ARINC 661

- MapWidget changed to MapHorz
- MapSource changed to MapHorz_Source
- MapItemList changed to MapHorz_ItemList

2.2.1 Definition Phase

This section modified for clarity.

2.2.3 Special Conditions

This section and subordinate sections added.

2.2.4 ARINC 661 Conformance

This section re-numbered.

2.2.5 ARINC 661 Library Evolution

This section re-numbered.

3.2.2 Widget Classification

Table 3.2.2-2 updated to support the expansion of widgets.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 updated to support the expansion of widgets.

3.2.5.3 Change Style Capabilities

Table 3.2.5.3 corrected error by replacing “Flashing” with “Animation”. Replace the definition with “Animation of ASCII text”.

3.2.5.5 Escape Sequences Description

Table 3.2.5.5.1 and 3.2.5.5.2 corrected error by replacing “Flashing” with “Animation”.

3.2.8 Map Management

This section modified for clarity.

3.3.1 Active Area

Added StyleSet Parameter.

3.3.2 BasicContainer

This section modified to state that some widgets can only be positioned at run-time. Supplement 1 supports the definition of the position of an optional panel at run-time. The objective is to define a position, not to move the widget at run-time.

3.3.5 CheckButton

This section modified to define the label alignment on button. The definition of the LabelPosition parameter is clarified. This parameter can be replaced by “PicturePosition” to be coherent with PictureXxxxButton.

3.3.6 ComboBox

This section modified to define the label alignment on button.

3.3.9 EditBoxMasked

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of ReportAllChanges changed from Boolean to Enumeration. STATE_CHANGE event deleted. ABORTED event added.

3.3.10 EditBoxNumeric

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of ReportAllChanges changed from Boolean to enumeration. STATE_CHANGE event deleted. ABORTED event added. Parameters NumericKeyFlag, MinValue, MaxValue and CyclicFlag are added. TicsCoarse and TicsFine are not modifiable at run-time.

3.3.11 EditBoxText

This section updated for clarity. The UA cannot change the EditBoxState into the edit mode through the EditBoxState parameter. The UA must request the Focus on the EditBox. When an EditBox reports all changes are completed and a crew member cancels the modifications, the display should annunciate an aborted event. Descriptive paragraph added. EditBoxState parameter deleted. Type of

ReportAllChanges changed from Boolean to enumeration. STATE_CHANGE event deleted. ABORTED event added.

3.3.20 Label

Changed “static” to “anonymous”. Deleted references to “blink” capability. Added “ColorIndex” parameter. Added additional Alignment value definitions.

3.3.21 LabelComplex

Added additional Alignment value definitions.

3.3.22 MapHorz_Item List

MapItemList changed to MapHorz_Item List.

3.3.22.1 MapHorz_ItemList Standard Items Description

Added “FilledPolyStart” and “FilledOval” map items.

3.3.22.2.1.8 Symbol Generic

SymbolType values were labeled as examples.

3.3.22.2.1.10 Symbol Rotated

SymbolType values were labeled as examples.

3.3.22.2.1.11 Symbol Runway

The words “coordinate of symbol” changed to “coordinate of threshold”.

3.3.22.2.1.12 FilledPolyStart

This section added.

3.3.22.2.1.13 SymbolOval

This section added.

3.3.23 MapLegacy

Parameter “FormatType” changed to “ChannelID”. All references to ARINC 702 and ARINC 708 were removed. This makes MapLegacy consistent in description and operation to ExternalSource widget.

3.3.24 MapHorz_Source

MapSource changed to MapHorz_Source. Table of MapDataFormat valued added for clarity. “A661_EVT_SELECTION” changed to “A661_EVT_SELECTION_MAP”.

3.3.25 MapHorz

This section modified to support map display. Map Widget changed to MapHorz. Description of “PRP Lat/Lng” identified as Commentary. Description of “Orientation” updated for clarity. Screen Reference Point X/Y changed from ulong to long.

3.3.28 PicturePushButton

Alignment parameter added.

3.3.29 PictureToggleButton

Alignment parameter added.

3.3.30 PopUpPanel

AutomaticClosure parameter added.

3.3.31 PopUpMenu

Replace “UAPositionFlag” by “OpeningMode”, because an UA may want to open a menu UP or DOWN.

3.3.32 PopUpMenuButton

It is necessary to define the label alignment on button. Replace “UAPositionFlag” by “OpeningMode”, because an UA may want to open a menu UP or DOWN.

3.3.33 PushButton

It is necessary to define the label alignment on button.

3.3.34 RadioBox

Updated to say that a user application may need to display a RadioBox without any selected element (for example, in the disable state).

3.3.36 ScrollPanel

HorizontalScroll and VerticalScroll modified to cover all possibilities, Absent/Up/Bottom/Left/right. For operational reasons, it might be necessary to place vertical and horizontal scroll at the same place. It is easier for a crew member to manage the scroll buttons.

3.3.37 ScrollList

It is necessary to define the label alignment on button.

3.3.38 Symbol

Category does not include “interactive”. This category was deleted.

3.3.39 TabbedPanel

It is necessary to define the label alignment on button. The UA managing a TabbedPanel (or a set of TabbedPanel) may need to define inset size. To introduce this functionality and keep the segregation between the TabbedPanel and the TabbedPanelGroup, new parameters were added. For Tabbed Panel, the “InsetSize” parameter is added. For TabbedPanelGroup, the “AutomaticInsetSizeFlag” parameter is added. This flag allows the choice between the manual inset size using “InsetSize” parameter or an inset size defined by a display dependent algorithm.

3.3.40 TabbedPanelGroup

The UA managing a TabbedPanel, or set of TabbedPanel, may need to define inset size. To introduce this functionality and to keep the segregation between TabbedPanel and the TabbedPanelGroup, new parameters were added. For Tabbed Panel, the “InsetSize” parameter is added. For TabbedPanelGroup, the “AutomaticInsetSizeFlag” parameter is added. This flag selects between the manual

inset size using “InsetSize” parameter, or an inset size defined by a display dependent algorithm.

3.3.41 ToggleButton

It is necessary to define the label alignment on button.

3.4 Widget Library Expansion

This section and its subordinate sections added by Supplement 1.

3.4.1 MapGrid

This section added. New Widget is defined.

3.4.2 ExternalSource

This section added. New Widget is defined.

3.4.3 MapVert

This section added. New Widget is defined.

3.4.4 MapVert_Source

This section added. New Widget is defined.

3.4.5 MapVert_ItemList

This section added. New Widget is defined.

3.4.6 EditBoXMultiLine

This section added. New widget is defined.

3.4.7 ComboBoxEdit

This section added. New widget is defined.

3.4.8 MenuBar

This section added. New widget is defined.

4.0 COMMUNICATION PROTOCOL

This section modified to reflect changes elsewhere in the document.

4.6 ARINC 661 Keyword Values

This section updated.

APPENDIX C – EXAMPLE OF A DEFINITION FILE

The example was updated following an actual implementation in 2003.

APPENDIX E - MAP MANAGEMENT TUTORIAL

This Appendix modified to provide example of a map display.

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

SUPPLEMENT 2
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM
INTERFACES TO USER SYSTEMS

Published: June 30, 2005

Prepared by the Airlines Electronic Engineering Committee

Adopted by the Airlines Electronic Engineering Committee: October 27, 2004

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to ARINC 661, Cockpit Display System Interface to User Systems. The supplement introduces seven new widgets that expand the capability of ARINC 661 display systems.

To make communication consistent across all widgets, changes to the communications protocol are included for some widgets, Standardized data structure and communication protocols are introduced, with the expectation that subsequent supplements will be compatible with this version. As a result of these changes, it is recognized that the communication protocol for some widgets included herein may not be compatible with communication protocols defined in earlier versions of this standard. The intent is for all CDS implementations built to this standard and future versions of this standard to use the same communication protocols.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document, vertical change bars in the margin will indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

0.0 Global Changes to Widgets and Data Structures Used in ARINC 661

The pad bits used within the data structures were modified to follow existing conventions more consistently.

The FocusIndex attribute was replaced by NextFocusWidget

ParameterStructure_XY was replaced by ParameterStructure_8Bytes

2.3.5 Cursor Management

This section was updated to provide guidance on cursor focus and highlight. The changes are intended to better describe cursor control timing.

3.1.3.5 Parameters Related to Focus Navigation

Table 3.1.3.5 was modified to identity of the widget to be focused upon was clarified.

3.1.4 Widget Events

This Section was added to describe behavior of widget events.

3.2.1 Widgets Summary

The widget library summary was expanded to include seven new widgets introduced in Supplement 2. These include:

1. MutuallyExclusiveContainer widget
2. ProxyButton widget

3. WatchdogContainer widget
4. Slider widget
5. PictureAnimated widget
6. SymbolAnimated widget
7. SelectionListButton widget

3.2.2 Widgets Classification

Table 3.2.2-2 was updated to support the widget expansion in Supplement 2.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 was expanded to include new widgets and the relationship with respect to each other.

3.3 Widget List

Table 3.3-1 was updated to correct the LSB value for 32 bit words. Conventions for padding and alignment were added.

3.3.1 ActiveArea

A parameter was modified to include the identity of the widget to focus upon.

3.3.2 BasicContainer

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.4 BufferFormat

The restrictions on BufferOf Parameter were modified.

3.3.4.1 A661_ParameterStructure_Buffer

This section was expanded to provide additional detail.

3.3.5 CheckButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.6 ComboBox

This section was modified to add new OpeningEntry parameter. A parameter was modified to include the identity of the widget to focus upon.

3.3.9 EditBoxMasked

An EntryValidation parameter was added. This would clarify use of alpha and numeric characters.

3.3.10 EditBoxNumeric

The FormatString parameter was changed to be run-time modifiable. New MaxFormatStringLength parameter was added. Events reported by this widget were updated.

3.3.11 EditBoxText

FormatString parameter was changed to be run-time modifiable. New MaxFormatStringLength parameter was added. Events reported by this widget were updated. New parameters StartCursorPos and LegendString were added.

3.3.13 GpArcCircle

Runtime parameters PosX and PosY were defined to be 8 bytes. UnusedPad were changed from 8 bits to 16 bits for proper alignment.

3.3.14 GpCrown

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.15 GpLine

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.16 GpLinePolar

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.17 GpRectangle

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.18 GpTriangle

Runtime parameters PosX and PosY were defined to be 8 bytes.

3.3.22.1 MapHorz_ItemList Standard Items Description

New MapItem types ITEM_SYNCHRONIZATION was added.

3.3.22.2 MapHorz_ItemList A661_ParameterStructure Specifics

The description of symbol placement was added. Also, throughout this section, RelativePosition attribute was added.

3.3.22.2.1.6 Line_Arc

This section was updated to clarify the description of InboundCourse and CourseChange angles.

3.3.22.2.1.14 Item_Synchronization

A new MapItem type was added.

3.3.22.3 MapHorz_ItemList Interactive Items

This section was added to define interactivity on maps.

3.3.24 MapHorz_Source

The EventFlag parameter was modified to support scroll wheels. A column was added to Table 3.3.24-2b to show direction of positive orientation for angles (clockwise or counter-clockwise).

3.3.25 Map_Horz

A new event was defined for Item Synchronization.

3.3.28 PicturePushButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.29 PictureToggleButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.30 PopUpPanel

A restriction was removed. PopUpPanels can be nested.

3.3.31 PopUpMenu

PictureArray was added as run-time modifiable parameter in Table 3.3.31-4.

3.3.32 PopUpMenuButton

A parameter was modified to include the identity of the widget to focus upon. The PictureArray parameter was added

3.3.33 PushButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.34 RadioBox

Restrictions were removed from this widget.

3.3.35 RotationContainer

The Enable parameter was deleted as it does not apply to this widget

3.3.36 ScrollPanel

The Horizontal Scroll parameter was modified to include Top/Bottom scroll. The Vertical Scroll parameter was modified to include Left/Right scroll.

3.3.37 ScrollList

The EnableArray parameter was added.

3.3.37.1 ScrollList Specific A661_ParameterStructure

New Section added.

3.3.39 TabbedPanel

A parameter was modified to include the identity of the widget to focus upon.

3.3.41 ToggleButton

A parameter was modified to include the identity of the widget to focus upon.

3.3.42 TranslationContainer

The Enable parameter was deleted as it does not apply to this widget. Runtime parameters TranslationX and TranslationY were defined to be 8 bytes.

3.4.1 MapGrid

The description of MapGrid parameters, IncrementX and IncrementY, were modified in Table 3.4.1-1. The changes would distinguish between distance-distance on a rectangular grid versus true bearing-distance grid used with weather radar. Support for various MapDataFormat values would be implementation dependent.

3.4.3 MapVert

Vertical display capability is included for terrain profile and other applications. A new event was defined for Item Synchronization. This would ensure that weather radar display and terrain display would be synchronized with aircraft position. Parameters RangeX and RangeY were retained.

3.4.4 MapVert_Source

The MapDataFormat parameter described in Table 3.4.4-2b was modified to allow the origin of the map to be an absolute geometric point or a point relative to a geometric reference point. A clarification was made to the use of different MapDataFormats in this widget.

3.4.5 MapVert_ItemList

Item Synchronization was added to Table 3.4.5.1-1.

3.4.5.2.1.10 Item_Synchronization

New Section added.

3.4.5.2.11 Symbol_Rotated

New Section added.

3.4.5.3 MapVert_ItemList Interactive Item

New Section added.

3.4.6 EditBoxMultiline

Changes made to event reporting of this widget.

3.4.7 ComboBoxEdit

A parameter was modified to include the identity of the widget to focus upon. Changes made to event reporting and entry validation.

3.5 Widget Library Expansion

This section and its subordinate sections were added. Seven widgets are included in the widget library expansion in Supplement 2.

3.5.1 MutuallyExclusiveContainer widget

MutuallyExclusiveContainer widget was added to activate a single widget from a collection of widgets.

3.5.2 ProxyButton widget

ProxyButton widget was added to direct a physical button push as an event selection on a window.

3.5.3 WatchdogContainer widget

WatchdogContainer widget was added to monitor the refresh rate of data supplied to the display system.

3.5.4 Slider widget

Slider widget was added to enable scrolling through the contents of a specific viewing area.

3.5.5 PictureAnimated widget

PictureAnimated widget was added to display of a set of bitmap images in rapid succession.

3.5.6 SymbolAnimated widget

SymbolAnimated widget was added to animate vector symbols.

3.5.7 SelectionListButton widget

SelectionListButton widget was added to allow a crew member to select one entry within a list.

4.5.1 Notation

Notation was corrected to specify [<A>] means zero or one of [<A>].

4.5.3.1 UADF loading structure

This section was added to describe the software data loading of the User Application Definition File (UADF).

4.5.3.2 Definition File (DF) Structure

The OEM File Header was replaced with the ARINC 665 definition file header, part of which can be implementation dependent. The block structure for Symbol Graphical Definition (SGD) was added.

4.5.3.5 Definition Time Symbol Block Commands

This section was added to address the symbol graphical definition (SGD) language.

4.5.3.6 Symbol Command Structure

This section added to address the symbol graphical definition (SGD) language.

4.5.3.7 Constraints Inside a Symbol Definition Block

This section was added to introduce Symbol Graphical Definition (SGD).

4.5.4.5.7 A661_Parameter_Structure_EnableArray

This section was added to support new widgets.

4.6 ARINC 661 Keyword Values

Section 4.6 was updated to include new constants, keywords and associated values.

5.0 Symbol Graphical Definition (SGD)

Symbol Graphical Definition (SGD) was added to define complex symbology.

6.0 XML Definition File Specification

This section was added as a suggested encoding method for ARINC 661 Definition Files.

Appendix C Example of a Definition File

This Appendix was updated to support the XML Definition File Specification added in Section 6. Section C.5 was added to provide an example XML definition file with a wide range of properties. Section C.6 was added as an example binary definition file that contains two symbol definitions. Section C.7 is the XML encoding of the binary definition file provided in Section C.6.

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

SUPPLEMENT 3
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM
INTERFACES TO USER SYSTEMS

Published: November 15, 2007

Prepared by the Airlines Electronic Engineering Committee

Adopted by the Airlines Electronic Engineering Committee: September 26, 2007

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to ARINC 661, Cockpit Display System Interface to User Systems. The supplement introduces six new widgets that expand the capability of ARINC 661 display systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In the past, changes introduced by a supplement to an ARINC Standard were identified by vertical change bars with an annotation indicating the change number. Electronic publication of ARINC Standards has made this mechanism impractical.

In this document, vertical change bars in the margin will indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

Global changes to ARINC 661 introduced by Supplement 3:

For reasons of conciseness, global changes introduced by this supplement are summarized here. These changes are repetitive changes that affect multiple sections in the same way.

An improvement to the handshake protocol between the CDS and the UA was introduced for validating pilot entries to the CDS. As part of this UA Validation change, the Enable parameter was redefined to be a three-state enumerated data type instead of a Boolean flag. Widget Parameter Tables, Creation Structure Tables and Runtime-Modifiable Parameter Tables were modified accordingly.

Other global changes:

- Added missing Type column to several Event Structure Tables.
- Added missing Size column to several Runtime-Modifiable Parameter Tables.
- Removed Values column from the four Runtime-Modifiable Parameter Tables that had this column: the information is found in the Widget Parameter Tables.
- Added notes to several widget description sections to clarify cases in which the cursor collides with widgets or layers which overlap one another.

Numerous minor editorial changes were applied to the document to improve:

- Table and Figure numbering consistency
- Text and Table formatting consistency
- Clarity and Grammar of Text descriptions.

2.2.4 ARINC 661 Conformance

This section was updated to better describe ARINC 661 compliance.

A reference to Appendix G was added to describe how new widgets can be introduced in ARINC 661.

2.3.5.2 From CDS to UA

Added clarification describing the behavior of overlapping widgets.

3.1.2.1 Widget States Definition

The section was expanded by the inclusion of Figure 3.1.2 (formerly included in Section 3.1.2.2).

3.1.2.2 Inner State Management: “Race Condition”

This section was updated to describe mitigation of race conditions. The reference to Appendix D was removed. (Appendix D was deleted by Supplement 3)

3.1.3.5 Parameters Related to Focus Navigation

This Section was modified to support FocusLink widget introduced in Section 3.6.4.

3.1.4 Widget Events

Widget Event Use Cross Reference Table 3.1.4-1 was added.

3.2.1 Widgets Summary

The widget library summary was expanded to include six new widgets introduced in Supplement 3. These include:

3.6.1 EditBoxNumericBCD
3.6.2 CursorRef
3.6.3 CursorOver
3.6.4 FocusLink, FocusIn, FocusOut
3.6.5 SizeToFitContainer
3.6.6 ShuffleToFitContainer

3.2.2 Widgets Classification

Table 3.2.2-1 and 3.2.2-2 was updated to support the widget expansion in Supplement 3. New widget categories “utility” and “UA validation” were added.

3.2.3.1 Possible Children of Container Widgets

Table 3.2.3.1 was expanded to include new widgets and the relationship with respect to each other.

3.2.5.1 Available Character Set

Clarify character set.

3.2.8.4 Map Synchronization Number

New Section added.

3.2.9 UA Validation

This Section replaces the old Section 3.2.9 (Non-Classified Widgets) deleted by Supplement 3. UA Validation describes the handshake protocol between the CDS and the UA for validating pilot entries and selections.

3.3.4.1 Buffer Format Alignment

This section renamed. The content remains the same.

3.3.6 ComboBox

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete and what is displayed during entry validation.

3.3.9 EditBoxMasked

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.10 EditNumeric

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.11 EditBoxText

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.3.20 Label

Changed Font parameter so it is runtime-modifiable.

3.3.22 MapHorz_ItemList Parameters

The Map Item Synchronization number was added. Table 3.3.22-1 - MapHorz_ItemList Parameters was updated. Table 3.3.22-4 - MapHorz_ItemList Runtime Modifiable Parameters was updated. Made spelling of BufferOfMapItems ParameterIdent match Section 4.

3.3.22.1 MapHorz_ItemList Standard Items Description

New MapHorz_Items for Symbol Target and Triangle Strips/Fans were added to Table 3.3.22.1 - MapHorz_ItemList Standard Items Description.

3.3.22.2.1.15 Symbol_Target

New Section added.

3.3.22.2.1.16 Triangle Strip Map Item

New Section added.

3.3.22.2.1.17 Triangle Segment Map Item

New Section added.

3.3.22.2.1.18 Triangle Segment Double Map Item

New Section added.

3.3.22.2.1.19 Triangle End Map Item

New Section added.

3.3.22.2.1.20 Triangle End Double Map Item

New Section added.

3.3.22.2.1.21 Triangle Fan Start Map Item

New Section added.

3.3.25 MapHorz Parameters

The Map Item Synchronization number was added. Table 3.3.25-1 - MapHorz Parameters was updated. Table 3.3.25-3 - MapHorz Runtime Modifiable Parameters was updated.

3.3.27 Panel

The motion allowed parameter was added. The position and size parameters are run-time modifiable. Table 3.3.27-1, Panel Parameters, was updated. Table 3.3.27-2, Panel Creation Structure, was updated. Table 3.3.27-3, Panel Runtime Modifiable Parameters, was updated.

3.3.37 ScrollList

This section updated for clarity. Two parameters were added.

3.4.1 MapGrid

The Map Item Synchronization number was added. Table 3.4.1-1 - MapGrid Parameters was updated. Table 3.4.1-3 - MapGrid Run-Time Modifiable Parameters was updated

3.4.3 MapVert

The Map Item Synchronization number was added. Table 3.4.3-1 - MapVert Parameters was updated. Table 3.4.3-3 - MapVert Runtime Modifiable Parameters was updated.

3.4.5 MapVert_ItemList

The Map Item Synchronization number was added. Table 3.4.5-1 - MapVert_ItemList Parameters was updated. Table 3.4.5-3 - MapVert_ItemList Runtime Modifiable Parameters was updated.

3.4.5.1 MapVert_ItemList Standard Items Description

New MapVert_Items for Triangle Strips and Fans were added to Table 3.4.5.1 - MapVert_ItemList Standard Items Description. Made spelling of BufferOfItems ParameterIdent match new spelling adopted in Section 4.

3.4.5.2.1.12 Triangle Strip MapVert_Item

This section was added.

3.4.5.2.1.13 Triangle Segment MapVert_Item

This section was added.

3.4.5.2.1.14 Triangle Segment Double MapVert_Item

This section was added.

3.4.5.2.1.15 Triangle End MapVert_Item

This section was added.

3.4.5.2.1.16 Triangle End Double MapVert_Item

This section was added.

3.4.5.2.1.17 Triangle Fan Start MapVert_Item

This section was added.

3.4.6 EditBoxMultiLine

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.4.7 ComboBoxEdit

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete, what is displayed during entry validation and what is displayed upon an invalid entry.

3.5.6 SymbolAnimated

Renamed AnimationFlag and LoopFlag as AnimationType and LoopFlag, respectively, and changed tables to correctly describe these parameters as enumerated parameters.

3.5.7 SelectionListButton

Deleted Commentary regarding alternate ways for UAs to signal CDS that validation is complete and what is displayed during entry validation.

3.6 Widget Library Expansion

This section and its subordinate sections were added. Six widgets are included in the widget library expansion in Supplement 3.

3.6.1 EditBoxNumericBCD

This section was added to define a new widget.

3.6.2 CursorRef

This section was added to define a new widget.

3.6.3 CursorOver

This section was added to define a new widget.

3.6.4 Focus Navigation Widgets

This section and its subordinate sections were added to define three new widgets to allow focus motion between layers.

3.6.5 SizeToFitContainer

This section was added to define a new widget.

3.6.6 ShuffleToFitContainer

This section was added to define a new widget.

4.4.2 Error Notification

This section was updated to clarify CDS behavior in error conditions.

4.4.3.1 Request from UA to CDS

This section expanded to include the capability to set cursor on widget.

4.5.3.2 Definition File (DF) Structure

Added picture definitions. Table 4.5.3.2-2 was clarified.

4.5.4.3 Request Structure

Added Table 4.5.4.3-6 containing cursor on widget description.

4.5.4.5.9 A661_ParameterStructure_BufferOfItems

Added a reference and clarified that this structure is used for both MapHorz_ItemList and MapVert_ItemList.

4.5.4.5.10 A661_ParameterStructure_Buffer

Structure table was added.

4.6 ARINC 661 Keyword Values

This section was updated to include new constants, keywords and associated values.

Table 4.6-2 updated to include picture block.

Table 4.6-3B updated to include symbol size, symbol rectangular and symbol circular.

Table 4.6-5 updated to include request cursor on widget.

Table 4.6-7 updated to include new keyword values.

Table 4.6-8 updated to include new keyword values.

Table 4.6-9 updated to delete material pertaining to unused events.

Table 4.6-10 updated to include a True_With_Validation value for the redefined Enable parameter. This is part of the UA Validation change.

Table 4.6-11 updated to include new keyword values.

5.2 Symbol Definition Commands

Symbol definition commands were updated to define an optional sensitive area representation.

5.2.1.3 Sensitive Area

This section was added to describe cursor sensitive areas for symbols.

5.2.3.13 Size

This section was added to define a new symbol definition command which allows the size of a symbol representation to be set.

6.0 XML Definition File Specification

This section was expanded to support OEM extensions and guidance for representing non-printable characters.

Expanded to include sensitive area and picture graphical definition.

Expanded to include support for defining bitmap images in XML definition files.

7.0 Picture Graphical Definition

New section added. It describes how bitmap images can be described in binary definition files using data structures called picture blocks.

Appendix C – Example of Definition File

This Appendix was updated and expanded. Examples C-1 through C-7 were updated for clarity and to bring them in line with changes to widget parameters. Example C-8, Binary DF Example, was added. Example C-9, XML Example, was added. Both C-8 and C-9 are examples of using bitmaps in DFs.

Appendix D – Use of StyleSet for Inner State Depiction

This Appendix was deleted by Supplement 3.

Appendix F – Communication Transport Protocols

This Appendix was added by Supplement 3.

Appendix G – New Widget Guidelines

This Appendix was added by Supplement 3.

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7435

SUPPLEMENT 4
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM
INTERFACES TO USER SYSTEMS

Published: May 10, 2010

Prepared by the Airlines Electronic Engineering Committee

Adopted by the AEEC Executive Committee: March 31, 2010

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to ARINC 661, Cockpit Display System Interface to User Systems. The supplement introduces three new widgets that expand the capability of ARINC 661 display systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In this document, bold blue text identifies technical changes to the document changed by the current supplement only. Changes made in previous supplements are described in the “ABC” pages for that supplement.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

For reasons of conciseness, global changes introduced by this supplement are included in the document. Many of these changes are repetitive and affect multiple sections of the document in the same way.

2.2.5 Library Evolution

This section was updated to describe ARINC 661 change management and how new widgets may be defined. The guidelines of Appendix G will be used when proposing changes and additions to the Widget Library.

2.2.5.1 Deprecation of ARINC 661 Widgets and Features

This section was added to describe a changed management technique called “deprecation”.

2.3.2.1 Layer Graphical Definition

This section was updated to clarify how widget pop-ups may be clipped. This will apply to PopUpMenu and PopUpPanel widgets.

2.3.2.3 Layer Priority Management

Section 2.3.2.3, Layer Priority Management, was updated to clarify how widgets should be drawn with respect to each other. Some widgets should always be drawn on top of other widgets. This will apply to PopUpMenu and PopUpPanel widgets.

3.1.1 Widget Identification

This section was updated with commentary to explain the role of the CDS with regard to User Application (UA) identification.

3.1.3.2 States of a Widget

The state “Interstate” was deleted from the list of state parameters.

3.1.4 Widget Events

Table 3.1.4, Widget Event Cross Reference, was updated to describe the possible states of PopUpPanelButton, SymbolPushButton, SymbolToggleButton widgets.

3.2.1 Widgets Summary

Section 3.2.1, Widgets Summary, was expanded to include three new widgets introduced by Supplement 4.

- SymbolPushButton widget
- SymbolToggleButton widget
- PopUpPanelButton widget

The description of ComboBox, Connector, PopUpPanel, PopUpMenu, MenuBar, ComboBoxEdit was clarified.

3.2.2 Widgets Classification

In Section 3.2.2, Widgets Classification, Table 3.2.2-1 and 3.2.2-2 was updated to support the widget expansion in Supplement 4.

3.2.3.1 Possible Children of Container Widgets

Section 3.2.3.1, Possible Children of Container Widgets, was expanded to include three new widgets and the relationship of each.

3.2.5 Text Strings

This section was updated to better describe “DefaultStyleText” and special formatting that may be used by CDS suppliers.

3.2.8.1 Horizontal Map Management

This section was updated for clarity. A MapHorz_ItemList contains a list of items to be drawn. Each item inside the MapHorz_ItemList can be modified at run-time which makes the list dynamic.

3.3 Widget List

Section 3.3, Widget List, describes the characteristics and the interface of each of the ARINC 661 widgets. This section was clarified to ensure that the definition of each widget is divided into specific parts as follows:

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

The fractional notation (fr) type was clarified to describe a scaled integer in Two’s Complement Fractional Notation. A clarification was made to state that indices will begin with the number 1 unless otherwise stated.

3.3.6 ComboBox

This section was updated to describe the operation of the ComboBox widget and to clarify how it may be clipped.

3.3.7 Connector

The description of this widget was updated in its entirety. This widget enables one layer to be the child of a container located in another layer. The changes expand the list of available parents to the Connector.

3.3.12 GpArcEllipse

This section was updated to correct the data type of the SizeX/SizeY parameter.

3.3.17 GpRectangle

This section was updated to correct the data type of the SizeX/SizeY parameter.

3.3.20 Label

Editorial changes were made to the description of this widget.

3.3.21 LabelComplex

Editorial changes were made to the description of this widget.

3.3.22 MapHorz_ItemList

This section and its subordinate sections were updated. The MapHorz_ItemList standard items description was updated to better describe use of EndFlag, Filled_Poly_Start, Item_Style, Legend, Legend_Anchor, and so forth. Interactive Triangle Strips and Fans were added. Three new Map_Item_List (MIL) items were added, Legend_Anchor_Rotated, Symbol_Rectangle and Draw_Line_to_Cursor. Relative position Map Symbology was clarified. Added a section to describe overlapping areas. The description of map item X/Y parameter data types were updated to refer to the map format data table in Section 3.3.24.

3.3.23 MapLegacy

The MapLegacy widget was deprecated by Supplement 4.

3.3.24 MapHorz_Source

This section was expanded by the addition of MDF_DIST_DIST_FIX data format, in which the coordinates are relative to a reference point other than the Projection Reference Point (PRP) or aircraft position. Optional run-time parameters were added to support the new format.

3.3.27 Panel

This section was updated to correct the data type of the SizeX/SizeY parameter.

3.3.29 PictureToggleButton

This section was updated to clarify use of the alternate flag parameter.

3.3.30 PopUpPanel

The description of this widget was updated to clarify the operation of the PopUpPanel widget and to clarify how it may be clipped. The PopUpPanel may be clipped to adjust the CDS look and feel.

3.3.31 PopUpMenu

The description of this widget was updated to clarify the operation of the PopUpMenu widget and to clarify how it may be clipped. The PopUpMenu may be clipped to adjust the CDS look and feel. The ParameterStructure_PictureArray data

structure was deprecated and replaced by ParameterStructure_2ByteArray. The ParameterStructure_EnableArray data structure was deprecated and replaced by ParameterStructure_1ByteArray.

3.3.32 PopUpMenuButton

The description of this widget was updated to clarify the operation of the PopUpMenuButton widget and to clarify how it may be clipped. The PopUpMenuButton may be clipped to adjust the CDS look and feel. The ParameterStructure_PictureArray data structure was deprecated and replaced by ParameterStructure_2ByteArray. The ParameterStructure_EnableArray data structure was deprecated and replaced by ParameterStructure_1ByteArray.

3.3.37 ScrollList

This section and its subordinate sections were updated. The ParameterStructure_EnableArray data structure was deprecated and replaced by ParameterStructure_1ByteArray. The contents of Section 3.3.37.1 ParameterStructure_EntryArray were moved to Section 4.5.4.5.17 and renamed to ParameterStructure_EntryListArray.

3.3.41 ToggleButton

This section was updated to clarify use of the alternate flag parameter.

3.4.1 MapGrid

This section was expanded by the addition of MDF_DIST_DIST_FIX data format, in which the coordinates are relative to a reference point other than the Projection Reference Point (PRP) or aircraft position. Optional run-time parameters were added to support the new format.

3.4.4 MapVert_Source

This section was clarified the use of map data formats for MapVert_Source.

3.4.5 MapVert_ItemList

This section and its subordinate sections were updated. A MapVert_ItemList contains a list of Items to be drawn. The types of map items inside can be modified at run-time which makes the list dynamic. The MapVert_ItemList standard items description was updated to better describe use of EndFlag, Filled_Poly_Start, Item_Style, Legend, Legend_Anchor, and so forth. Interactive Triangle Strips and Fans were added. Three new Map_Item_List (MIL) items were added, Legend_Anchor_Rotated, Symbol_Target and Draw_Line_to_Cursor. Relative position Map Symbology was clarified. Added a section to describe overlapping areas. The description of map item X/Y parameter data types were updated to refer to the map format data table in Section 3.3.24.

3.4.7 ComboBoxEdit

This section was updated to describe the operation of the ComboBoxEdit widget and to clarify how it may be clipped.

3.4.8 MenuBar

This section was updated to describe the list of children to the MenuBar widget. It was updated to say that a group of buttons may have their sizes and positions

managed by the CDS in a consistent manner. The data part of the ButtonSize parameter was corrected.

3.5.7 SelectionListButton

This section was updated to describe the operation of the ComboBoxEdit widget and to clarify how it may be clipped.

3.6.3 CursorOver

This section was updated to correct the data type of the SizeX/SizeY parameter.

3.6.4.2 FocusIn

The description of this widget was updated. The FocusIn widget is used together with the FocusOut widget to define a NextFocusedWidget sequence that crosses between two separate layers. The connection between the two layers is performed via FocusInRef, an integration level parameter managed by the CDS. The FocusOut widget's NextFocusInRef parameter contains the FocusInRef value identifying the FocusIn widget that will receive the focus.

3.6.4.3 FocusOut

The description of this widget was updated. The FocusOut widget is used together with the FocusIn widget to define a NextFocusedWidget sequence that crosses between two separate layers. The connection between the two layers is performed via FocusInRef, an integration level parameter managed by the CDS. The NextFocusInRef parameter contains the FocusInRef value of the FocusIn widget that will receive the focus when the focus reaches the FocusOut widget.

3.6.5 SizeToFitContainer

This section was expanded to include SymbolPushButton and SymbolToggleButton to the list of children of this widget.

This section was updated to correct the data type of the SizeX/SizeY parameter.

3.6.6 ShuffleToFitContainer

This section was expanded to include SymbolPushButton and SymbolToggleButton to the list of children of this widget.

This section was updated to correct the data type of the NumberOfVisibleChildren parameter.

3.7 Widget Library Expansion - Supplement 4

Section 3.7, Widget Library Expansion, and its subordinate sections were added to describe three new widgets were introduced by Supplement 4.

3.7.1 SymbolPushButton

This section added by Supplement 4.

3.7.2 SymbolToggleButton

This section added by Supplement 4.

3.7.3 PopUpPanel Button

This section added by Supplement 4.

4.5.4.5.7 A661_ParameterStructure_EnableArray

This structure was deprecated by Supplement 4. It was replaced by A661_ParameterStructure_1ByteArray.

4.5.4.5.11 A661_ParameterStructure_1ByteArray

Formerly titled “A661_ParameterStructure_EntryPopUpArray,” this section was completely re-written for Supplement 4.

4.5.4.5.12 A661_ParameterStructure_1ByteArray_CellStructure

This section added by Supplement 4.

4.5.4.5.13 A661_ParameterStructure_2ByteArray

This section added by Supplement 4.

4.5.4.5.14 A661_2ByteArray_CellStructure

This section added by Supplement 4.

4.5.4.5.15 A661_ParameterStructure_EntryPopUpArray

This section added by Supplement 4.

4.5.4.5.16 A661_EntryPopUpStructure

This section added by Supplement 4.

4.5.4.5.17 A661_ParameterStructure_EntryListArray

4.6 ARINC 661 Keyword Values

This section was updated to include new constants, keywords and associated values.

5.1 Overview

This section was expanded to include Symbol_Target and revised for clarity.

6.2 Description

This section was updated to include a new rule for using XML. The spelling of XML attribute names must match the spelling of widget parameter names.

6.2.3 Properties

This section was updated to explain how to represent illegal XML characters in XML.

Appendix A – Glossary

Editorial changes were made to several glossary terms.

Appendix C – Example of Definition File

An example was added to describe ARINC 661 Escape Sequences. Minor corrections were made to existing examples.

Appendix E – Map Management Tutorial

This appendix was updated to describe the proper use of MapItemList (MIL) End_Flags.

Appendix G – Guidelines for the Introduction of New Widget Guidelines

This appendix was expanded to describe how widgets and widget elements may be deprecated. Guidelines for naming widget parameters were added. Guidelines for numbering array indices were added.

Appendix H – CDS Layer and Window Management

This section was added to describe a super-layer concept that defines the relationship between all layers that can be drawn on a CDS.

Appendix I – ARINC 661 API Design Issues

This section was added to describe ARINC 661 Application Program Interface (API) considerations.

AERONAUTICAL RADIO, INC.
2551 Riva Road
Annapolis, Maryland 24101-7435

SUPPLEMENT 5
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

Published: July 1, 2013

Prepared by the AEEC

Specification 661-5 Adopted by the AEEC Executive Committee: April 24, 2013

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to **ARINC 661:** *Cockpit Display System Interface to User Systems*. The supplement introduces widget extensions, which add features to existing widgets without disturbing the existing installed base. In addition, ten new widgets are introduced that expand the capability of ARINC 661 display systems. Twelve additional Greek characters are added to the character set. Chinese and Cyrillic alphabet capabilities are also incorporated.

B. ORGANIZATION OF THIS SUPPLEMENT

In this document, **bold blue** text identifies technical changes to the document changed by the current supplement only. Changes made in previous supplements are described in the “ABC” pages for that supplement.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

For reasons of conciseness, global changes introduced by this supplement are included in the document. Many of these changes are repetitive and affect multiple sections of the document in the same way.

2.2.5.2 Widget Extension

This section is added to describe widget extensions.

2.3.2.4 Layer Activity/Visibility Management

Corrected reference to re-named Section 4.4.3.2 from “Request/Notification from CDS to UA” to “Notification from CDS to UA”.

2.3.2.4.1 Visibility

Added clarifying language to this section.

2.3.2.4.2 Activity

Added clarifying language to this section and revised commentary.

2.3.2.5 Layer Context Management

Added clarifying language to this section.

3.1.2.2 Inner State Management: “Race Condition”

Added clarifying language to this section.

3.1.3.3 Look and Feel Characteristics of a Widget: “Styleset” and “Halo” Parameters

Added Halo parameter to the styleset to enable full outline of a widget and enhance readability.

3.1.3.5 Parameters Related to Focus Navigation

Added language describing CDS focus sequence when encountering invisible or disabled widgets.

3.1.4 Widget Events

Due to increasing size, Table 3.1.4 was split into Tables 3.1.4-1 and 3.1.4-2 illustrating different events. Added A661_EVT_OFFSCREEN and A661_661_EVT_ONSCREEN events to Table 3.1.4-1.Both tables were updated to include MapHorz_Container and Paging Container.

3.2.1 Widget Summary

Updated Table 3.2.1 Widget Library Summary to delete the MapLegacy widget. Broadcast Receiver, DataScalingFR180, DataScalingLong, DataScalingULong, GPPolyline, MapHorz_Container, MapHorz_Panel, NoServiceMonitor, NumericReadout, and PagingContainer added as new widgets for Supplement 5.

3.2.2 Widget Classification

Updated Table 3.2.2-2 Widget Classification Table to delete the MapLegacy widget, and add dynamic motion capability to BasicContainer, Panel, MutuallyExclusive Container, CursorRef, and CursorOver widgets. Also included new widgets GpPolyline, PagingContainer, NumericReadout, MapHorz_Container, MapHorz_Panel, DataScalingLong, DataScalingULong, DataScalingFR180, BroadcastReceiver, and NoService Monitor as new widgets added for Supplement 5.

3.2.3.1 Possible Children of Container Widgets

Due to increasing size, Tables 3.2.3.1-1 and 3.2.3.1-2 were merged into a single landscape formatted table. The table was updated to remove the MapLegacy widget (deleted) and add MapHorz_Container, MapHorz_Panel, NoServiceMonitor, and PagingContainer as possible parent widgets and BroadcastReceiver, DataScalingLong, DataScalingULong, DataScalingFR180, GpPolyline, MapHorz_Container, MapHorz_Panel, NoService Monitor, NumericReadout, and PagingContainer as new widgets added for Supplement 5.

3.2.5.1 Available Character Set

The following twelve Greek letters are added to the available character set for Supplement 5: α(alpha), β (beta), γ (gamma), φ (phi), θ (theta), ψ (psi), ρ (rho), ε(epsilon), π(pi), ω (omega), Δ(delta), and τ(tau). These comprise an intentional mixture of upper and lower case letters selected according to their relevance and common usage in aviation.

3.2.5.3 Change Style Capabilities.

The Repeat character was renamed as the Occurrence character and redefined as the number of times the set of characters embedded between the escape sequences should appear on the display. A note was added advising that the use of zero-value P1, Tvalue1, and Tvalue2 parameters may complicate implementations.

3.2.5.5 Escape Sequences Description

Table 3.2.5.5-1 was adjusted to reflect that the Repeat character was renamed as the Occurrence character and redefined as the number of times the set of characters embedded between the escape sequences should appear on the display. The size column was also deleted. Clarifying notes and commentary were also added.

3.2.5.6 Text Alignment

This section added to describe and illustrate where text should be placed relative to the anchor position.

3.2.9 UA Validation

Clarifying commentary added.

3.2.10 Widget Extension

This section was added to alert system designers of new capabilities available through the use of widget extensions.

3.3 Widget List

Added clarifying language. Modified Table 3.3-1 to correct uchar parameter description, enhance string parameter description and add short parameter.

3.3.1 ActiveArea

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.1-1 and Table 3.3.1-4.

3.3.2 BasicContainer

Added dynamic motion to the BasicContainer widget and clarified that the Enable parameter applies to this widget's descendants.

3.3.5 CheckButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.5-1 and Table 3.3.5-4.

3.3.6 ComboBox

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.6-1 and Table 3.3.6-4.

3.3.7 Connector

Clarified that the Enable parameter applies to this widget's descendants.

3.3.9 EditBoxMasked

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.9-7.

3.3.10 EditBoxNumeric

Enhanced description of how the ReportAllChanges parameter operates with respect to EditBoxNumeric. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.10-1 and Table 3.3.10-7.

3.3.11 EditBoxText

Enhanced description of how the ReportAllChanges parameter operates with respect to EditBoxText. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.11-1 and Table 3.3.11-7.

3.3.12 GpArcEllipse

Added haloing at the same level in Table 3.3.12-2.

3.3.13 GpArcCircle

Enhanced the Halo parameter description in Table 3.3.13-1 and added A661_Same_Level as a value/range option in Table 3.3.13-2.

3.3.14 GpCrown

Enhanced the Halo parameter description in Table 3.3.14-1 and added A661_Same_Level as a value/range option in Table 3.3.14-2.

3.3.15 GpLine

Enhanced the Halo parameter description in Table 3.3.15-1 and added A661_Same_Level as a value/range option in Table 3.3.15-2.

3.3.16 GpLinePolar

Enhanced the Halo parameter description in Table 3.3.16-1 and added A661_Same_Level as a value/range option in Table 3.3.16-2.

3.3.17 GpRectangle

Enhanced the Halo parameter description in Table 3.3.17-1 and added A661_Same_Level as a value/range option in Table 3.3.17-2.

3.3.18 GpTriangle

Enhanced the Halo parameter description in Table 3.3.18-1 and added A661_Same_Level as a value/range option in Table 3.3.18-2.

3.3.22 MapHorz_ItemList

Added BlockedBufferOfMapItems as a new parameter in Table 3.3.22-1 MapHorz_ItemList Parameters and Table 3.3.22-4 MapHorz_ItemList Runtime Modifiable Parameters.

3.3.22.1 MapHorz_ItemList Standard Items Description

Added Legend_Combo and Legend_Highlight to Table 3.3.22.1 as a standard items.

3.3.22.2.1.3 Legend, Legend_Highlight and Legend_Pop_Up

Added Legend_Highlight as a new item type.

3.3.22.2.1.8 Symbol_Generic

Enhanced Symbol_Generic description.

3.3.22.2.1.10 Symbol_Rotated

Enhanced Symbol_Rotated description.

3.3.22.2.1.25 Legend_Combo

Added this new item allowing fewer map items to be used defining a specific behavior.

3.3.22.2.3 A661_ParameterStructure_BlockedBufferOfItems

Added this new parameter to address update synchronization at the MapHorz_ItemList widget level.

3.3.23 MapLegacy (DELETED)

The MapLegacy widget was deprecated in Supplement 4, and deleted from the specification in Supplement 5.

3.3.27 Panel

Added dynamic motion category to the Panel widget.

3.3.28 PicturePushButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.28-1 and Table 3.3.28-4.

3.3.29 PictureToggleButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.29-1 and Table 3.3.29-4.

3.3.31 PopUpMenu

Noted in Table 3.3.31-4 that the EnableArray and PictureArray parameters were deprecated in Supplement 4.

3.3.32 PopUpMenuButton

Noted in Table 3.3.32-4 that the EnableArray and PictureArray parameters were deprecated in Supplement 4. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.32-1 and Table 3.3.32-4.

3.3.33 PushButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.33-1 and Table 3.3.33-4.

3.3.36 ScrollPanel

Added interactive category to the ScrollPanel widget. Modified description for improved clarity. Revised Figure 3.3.36 Frame Standard Coordinate System and included additional explanatory language. Revised description of HomeX, HomeY, FrameX and FrameY parameters in Table 3.3.36-1.

3.3.37 ScrollList

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.37-1 and Table 3.3.37-5.

3.3.39 TabbedPanel

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.39-1 and Table 3.3.39-3.

3.3.41 ToggleButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.3.41-1 and Table 3.3.41-4.

3.3.42 TranslationContainer

Updated description to better describe how widgets placed within a TranslationContainer are referenced.

3.4.5 MapVert_ItemList

Modified Table 3.4.5-1 to include BlockedBufferOfItems parameter.

3.4.5.1 MapVert_ItemList Standard Items Description

Table 3.4.5.1 was updated to add Legend_Combo, Legend_Highlight, and Symbol_Rectangle items to MapVert.

3.4.5.2.1.3 Legend, Legend_Highlight and Legend_Pop_Up

Added Legend_Highlight as new item type.

3.4.5.2.1.21 Symbol_Rectangle

This section was added to define a Symbol_Rectangle Map_Item_List (MIL) item for use in MapVert.

3.4.5.2.1.22 Legend_Combo

This section was added to define a Legend_Combo Map_Item_List (MIL) item for use in MapVert.

3.4.5.2.3 A661_ParameterStructure_BlockedBufferOfItems

This section added to define a BlockedBufferOfItems parameter.

3.4.5.3 MapVert_ItemListInteractiveMap Items

This section expanded to include an enhanced description and provide a summary listing of interactive map items.

3.4.6 EditBoxMultiLine

Revised description of how the ReportAllChanges parameter operates. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.4.6-1 and Table 3.4.6-7.

3.4.7 ComboBoxEdit

Revised description of how the ReportAllChanges parameter operates. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.4.7-1 and Table 3.4.7-8.

3.5.1 Mutually Exclusive Container

Added the Container category to this widget.

3.5.3 WatchdogContainer

Updated description of how child widgets are evaluated for display. Added commentary regarding initial state and evaluating period.

3.5.4 Slider

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.5.4-1 and Table 3.5.4-4.

3.5.7 SelectionListButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.5.7-1 and Table 3.5.7-4.

3.6.1 EditBoxNumericBCD

Revised description of how the ReportAllChanges parameter operates. Modified value/range of ReportAllChanges. Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.6.1-1 and Table 3.6.1-7.

3.6.2 CursorRef

Added Dynamic Motion to this widget.

3.6.3 CursorOver

Added Dynamic Motion to this widget.

3.6.4.2 FocusIn

Added Runtime modifiability to NextFocusedWidget and Table 3.6.4.2-3 FocusIn Runtime Modifiable Parameters.

3.6.4.3 FocusOut

Added layer, application, and focus reference parameters to Table 3.6.4.3-1 FocusOut Parameters and Table 3.6.4.3-2 FocusOut Creation Structure. Added Table 3.6.4.3-3 FocusOut Runtime Modifiable Parameters.

3.6.5 SizeToFitContainer

Added Dynamic Motion to this widget.

3.6.6 ShuffleToFitContainer

Added dynamic motion to this widget and modifed PosX, PosY, SizeX, and SizeY parameters to be runtime modifiable. Expanded Table 3.6.6-3 to include PosX, PosY, Size X, Size Y, and StyleSet as runtime modifiable parameters.

3.7.1 SymbolPushButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.7.1-1 and Table 3.7.1-4.

3.7.2 SymbolToggleButton

Modified NextFocusedWidget and AutomaticFocusMotion to be runtime modifiable, delineated in Table 3.7.2-1 and Table 3.7.2-4.

3.7.3 PopUpPanelButton

Added commentary regarding automatic focus motion and added NextFocusedWidget as a parameter in Table 3.7.3-1 PopUpPanelButton Parameters and Table 3.7.3-2 PopUpPanelButton Creation Structure.

3.8 Widgets added for Supplement 5.

This section added with subsections describing ten new widgets added for Supplement 5: GpPolyline, Paging Container, Numeric Readout, MapHorz_Container, MapHorz_Panel, DataScalingLong, DataScalingUlong, DataScalingFR180, BroadcastReceiver, and NoServiceMonitor.

3.8.1 GpPolyline

This section added to describe the GpPolyline widget.

3.8.2 PagingContainer

This section added to describe the PagingContainer widget.

3.8.3 NumericReadout

This section added to describe the NumericReadout widget.

3.8.4 MapHorz_Container

This section added to describe the MapHorz_Container widget.

3.8.5 MapHorz_Panel

This section added to describe the MapHorz_Panel widget.

3.8.6 DataScalingLong

This section added to describe the DataScalingLong widget.

3.8.7 DataScalingULong

This section added to describe the DataScalingULong widget.

3.8.8 DataScalingFR180

This section added to describe the DataScalingFR180 widget.

3.8.9 BroadcastReceiver

This section added to describe the BroadcastReceiver widget.

3.8.10 NoServiceMonitor

This section added to describe the NoServiceMonitor widget.

4.4.1 Type of Commands

Expanded to include definition-time commands for widget extensions.

4.4.2 Error Notification

Added Extend as a command to Table 4.4.2 Exception Type.

4.4.3.2 Notification from CDS to UA

Renamed from the former Request/Notification from CDS to UA. Added A661_NOTE_CYCLIC_ACTIVATION notification to Table 4.4.3.2 and renamed Notification from CDS to UA.

4.5.3.2 Definition File (DF) Structure

Added { A661_Charset_Encoding_Structure_DT } to the Definition File. Added commentary regarding layers, symbols and pictures. Added Table 4.5.3.2-6 Charset Encoding Block Structure Exchanged Between US and CDS at Definition Time. This was added to accommodate Cyrillic and Chinese characters.

4.5.3.3 Layer Block Definition Commands

Renamed from the former Definition Time Block Commands. Renamed Table 4.5.3.3 to Layer Block Definition Commands and expanded to include a definition command for extending widgets.

4.5.3.3.2 Extension Command Structure

This section added to define extension command structure.

4.5.3.3.3 Constraints Inside a UALD Block

Renumbered from the previous 4.5.3.4.

4.5.3.4 Symbol Block Definition Commands

Renumbered and renamed from the former 4.5.3.5 Definition Time Symbol Block Commands.

4.5.3.4.1 Create Symbol Command Structure

Renumbered and renamed from the former 4.5.3.6 Symbol Command Structure.

4.5.3.4.2 Constraints Inside a Symbol Definition Block

Renumbered from the former 4.5.3.7.

4.5.4.1 Run-Time Block Commands

Corrected syntax in Table 4.5.4.1-2 to indicate it can be one of several types.

4.5.4.3 Request Structure

Corrected syntax in Table 4.5.4.3-1 to indicate it can be one of several types.

4.5.4.4 Notification Structure

Corrected syntax in Table 4.5.4.4-1 to indicate it can be one of several types. Added A661_Cyclic_Activation_Struct to the layer notification structure to Table 4.5.4.4-1. Added reserved space for OEM customization in Table 4.5.4.4-2. Added Table 4.5.4.4-5 Cyclic_Activation_Data_Structure.

4.5.4.5.19 A661_ParameterStructure_8ByteArray

This section added to define 8 byte array parameter structure.

4.5.4.5.20 A661_ParameterStructure_8ByteArray_CellStructure

This section added to define 8 byte array cell structure.

4.5.4.5.21 A661_ParameterStructure_BlockedBufferOfItems

This section added to define BlockedBufferOfItems parameter structure.

4.6 ARINC 661 Keyword Values

Table 4.6-2 expanded to include character set encoding blocks. Table 4.6-3a Commands expanded to include a widget extension command. Table 4.6-3b expanded to include Symbol Definition Bounding _Circle and _Rectangle, End_ Group, Set_Rotation, Start_Group and Set_Translation. Former Table 4.6-5 Requests/Notifications split into Table 4.6-5a Notes with Cyclic Activation added and Table 4.6-5b Requests with Cursor_on_Widget, Focus_On_Widget, Layer_Active, Layer_Inactive and Layer_Visible added. Table 4.6-6 Exception Type expanded to include a widget extension aborted notice. Table 4.6-7 Widgets was updated to remove the MapLegacy widget, include previously omitted PopUpPanelButton, and new widgets Broadcast Receiver, GpPolyline, MapHorz_Container, MapHorz_Panel, No_Service_Monitor, Numeric_Readout, Paging_Container, Pop_Up_Panel_Button, Data_Scaling_Long, Data_Scaling_ULong, Data_Scaling_FR180 and reserved space. Updated table 4.6-8 Parameter Types, 4.6-9 Event Types, and 4.6-11 Integer Constant Values. Added new tables 4.6-12 Widget Extensions and 4.6-13 Character set encodings.

5.2.1.1 Focus

Assigned name and number to Table 5.2.1.1 FOCUS Command Structure.

5.2.1.2 Highlight

Assigned name and number to Table 5.2.1.2 HIGHLIGHT Command Structure.

5.2.1.3 Sensitive Area

Renumbered and renamed figure 5.2.1.3. Assigned name and number to Table 5.2.1.3-1 Sensitive Area-Rectangle Command Structure and Table 5.2.1.3-2 Sensitive Area-Circular Command Structure.

5.2.2.1 Set Color

Assigned name and number to Table 5.2.2.1 SET_COLOR Command Structure.

5.2.2.2 Set Line Style

Assigned name and number to Table 5.2.2.2 SET_LINE_STYLE Command Structure.

5.2.2.3 Set Font

Assigned name and number to Table 5.2.2.3 SET_FONT Command Structure.

5.2.2.4 Set Halo

Assigned name and number to Table 5.2.2.4 SET_HALO Command Structure. Added explanatory language.

5.2.3 Coordinate System commands

Added this new section and subsections to allow translation and rotation transformations to subsequent graphical commands.

5.2.4 Graphic Primitive Commands

Renumbered from the previous 5.2.3. Enhanced explanatory comments.

5.2.4.1 Legend Anchor

Renumbered from the previous 5.2.3.1. Assigned name and number to Table 5.2.4.1 LEGEND_ANCHOR Command Structure. Enhanced explanatory comments.

5.2.4.2 Arc Ellipse

Renumbered from the previous 5.2.3.2. Assigned name and number to Table 5.2.4.2 ARC_ELLIPSER Command Structure.

5.2.4.3 Arc Circle

Renumbered from the previous 5.2.3.3. Assigned name and number to Figure 5.2.4.3 Arc Circle and Table 5.2.4.3 ARC_CIRCLE Command Structure.

5.2.4.4 Crown

Renumbered from the previous 5.2.3.4. Assigned name and number to Figure 5.2.4.4 Crown and Table 5.2.3.4 CROWN Command Structure.

5.2.4.5 Line

Renumbered from the previous 5.2.3.5. Assigned name and number to Table 5.2.4.5 LINE Command Structure.

5.2.4.6 Line Polar

Renumbered from the previous 5.2.3.6. Assigned name and number to Table 5.2.4.6 LINE_POLAR Command Structure.

5.2.4.7 Polyline

Renumbered from the previous 5.2.3.7. Renumbered vertices and assigned name to Figure 5.2.4.7 Polyline and Table 5.2.4.7 POLYLINE Command Structure.

5.2.4.8 Rectangle

Renumbered from the previous 5.2.3.8. Assigned name and number to Table 5.2.4.8 RECTANGLE Command Structure.

5.2.4.9 Triangle

Renumbered from the previous 5.2.3.9. Assigned name and number to Table 5.2.4.9 TRIANGLE Command Structure.

5.2.4.10 Triangle Fan

Renumbered from the previous 5.2.3.10. Assigned names and numbers to Figures 5.2.4.10-1 Triangle Fan Concave Polygon, 5.2.4.10-2 Triangle Fan Convex Polygon, and Table 5.2.4.10 TRIANGLE_FAN Command Structure.

5.2.4.11 Triangle Strip

Renumbered from the previous 5.2.3.11. Assigned name and number to Figure 5.2.4.11 Triangle Strip and Table 5.2.4.11 TRIANGLE_STRIP Command Structure.

5.2.4.12 Text

Renumbered from the previous 5.2.3.12. Assigned name and number and enhanced descriptions in Table 5.2.4.12 TEXT Command Structure.

5.2.4.13 Size

Deprecated in Supplement 5 and superseded by Bounding_Rect and Bounding_Circle commands. Renumbered from the previous 5.2.3.13. Assigned name and number to Table 5.2.4.13 SIZE Command Structure.

5.2.4.14 Bounding Rectangle

Added this new section describing the BOUNDING_RECT command which defines the bounding box of a symbol representation.

5.2.4.15 Bounding Circle

Added this new section describing the BOUNDING_CIRCLE command which defines the bounding circle of a symbol representation.

6.2.1 Charset Encoding

This section added to enable character set encoding. This was added to accommodate Cyrillic and Chinese characters.

6.2.2 Picture and Symbol Graphical Definitions

Renumbered from the previous 6.2.1.

6.2.3 Layers and Widgets

Renumbered from the previous 6.2.2. Added table describing XML extension elements.

6.2.4 Properties

Renumbered from the previous 6.2.3. The Repeat character was renamed as the Occurrence character.

6.3 Document Type Definition (DTD) Specification

Inserted DTD for character set encoding.

7.2 Picture Definition Structures

Renamed Table 7-1 Picture Block Structure Exchanged Between UA and CDS at Definition Time.

8.0 Widget Extensions

This section is added with subsections to describe the rules for and use of widget extensions as well as to define possible widgets for extension. Widget extensions can be utilized to add new optional features to an existing widget without breaking backward compatibility. Multiple extensions can be applied to a single widget to add multiple optional features. Runtime modifiable parameters and events may also be associated with an extension.

Appendix A – GLOSSARY

Definition of Identifier was added. Editorial changes were made to Reference.

Appendix C – EXAMPLE OF A DEFINITION FILE

C-2 Example of Application Code at Run Time

Commentary added.

C-6 An XML DF Example with extensions

This section added to illustrate an XML file example using extensions on widgets.

C-7 Binary DF Specifying Symbol Graphical Definitions

This section renumbered from the previous C-6.

C-8 XML DF Specifying Symbol Graphical Definitions

This section renumbered from the previous C-7.

C-9 Binary DF Specifying Picture Definitions

This section renumbered from the previous C-8.

C-10 XML DF Specifying Picture Definitions

This section renumbered from the previous C-9.

C-11 Binary DF Specifying Escape Sequences

This section renumbered from the previous C-10.

C-12 XML DF Specifying DF Escape Sequences

This section renumbered from the previous C-11.

Appendix F – COMMUNICATION TRANSPORT PROTOCOLS

F-6 Example Protocol for Reliable Communications

Assigned name and number to Figure F-6 Extended Block Data Structure.

Appendix G – NEW WIDGET GUIDELINES

G-2 Description/Discussion

Added language advising designers that Widget Extensions should be considered as an alternative to adding a new widget.

G-5 Widget Category

Assigned name and number to Table G-5 Widget Parameter Table Template.

G-6 Commonly Used Parameters List

Assigned name and number to Table G-6 Commonly Used Widget Parameters.

G-7 Creation Structure

Added language for pads and assigned name and number to Table G-7 Creation Structure Table Template. Clarified that all unused pads are to be set to zero.

G-8 Event Structure

Assigned name and number to Table G-8 Event Structure Table Template and modified to value/description to reflect standard to be utilized for an unused pad.

G-9 Runtime Modifiable Parameters

Assigned name and number to Table G-9 Runtime Modifiable Parameter Structure Table Template.

G-10 Widget Library Tables

Added bullet point referencing possible widgets for extension.

G-11 Modification Property of Parameters

Added this section and subsections providing an explanation and guidelines for exceptions to the general rule that all parameters should be definition and runtime modifiable.

Appendix H – CDS LAYER AND WINDOW MANAGEMENT

H-5 Window Management

Added commentary regarding architectures for window management.

H-6.2 DataConnector

Added bullet point referencing Connector widget restrictions.

Appendix I – ARINC 661 UA DESIGN CONSIDERATIONS

Renamed from the previous ARINC 661 API Design Issues.

This appendix rewritten for Supplement 5.

Appendix J – LOOK MODELING

This appendix added to define graphical characteristics of widget appearance such as color and border properties.

Appendix K – UA/CDS INTERFACE

This appendix added to define the UA/CDS interface and illustrate the flow of commands and notifications used by the UA at runtime.

SAE INDUSTRY TECHNOLOGIES CONSORTIA (SAE ITC)
16701 Melford Blvd., Suite 120
Bowie, Maryland 20715 USA

SUPPLEMENT 6
TO
ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACES TO USER SYSTEMS

Published: September 1, 2016

Prepared by the AEEC

Specification 661-6 Adopted by the AEEC Executive Committee: April 26, 2016

A. PURPOSE OF THIS DOCUMENT

This supplement introduces numerous changes and additions to **ARINC Specification 661:** *Cockpit Display System Interface to User Systems*. This Supplement introduces seven widgets, three widget extensions, touchscreen, and three dimensional vision capabilities to ARINC 661 display systems.

B. ORGANIZATION OF THIS SUPPLEMENT

In this document, **bold blue** text identifies technical changes to the document changed by the current Supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

For reasons of conciseness, global changes introduced by this supplement are included in the document. Many of these changes are repetitive and affect multiple sections of the document in the same way.

1.2 Relationship to Other Documents

Added clarifying language that the latest versions of certain other ARINC Standards apply to the development of a CDS.

1.8 Applicability

Renumbered the previous Section 1.9 to Section 1.8. Removed the previous Section 1.8 Reference Documents and consolidated into Section 1.2, Relationship to Other Documents.

1.9 Associated Electronic Files

Added this new section delineating the electronic support files, links to them, and related copyright and disclaimer information. XSD schema, URI, version, and location information is also provided.

2.1 Introduction

Included additional input devices to cover the use of touch screen technology.

2.3.5 Cursor Management

Added language to describe cursor management with touch screen technology.

2.3.5.2 From CDS to UA

Added commentary language for touch screen use cases.

2.3.6 Touch Screen Management

Added this section to describe CDS to UA communication through widgets.

3.1.1 Widget Identification

Added clarifying language to the commentary.

3.1.4 Widget Events

Expanded Table 3.1.4-1 Widget Event Cross Reference with the following new events: GESTURE_DIR_FLICK, GESTURE_DOUBLE_TAP, GESTURE_DRAG, GESTURE_FLICK, GESTURE_PINCH, GESTURE_PRESS_AND_HOLD,

GESTURE_ROTATE, GESTURE_TAP, EVT_ITEM_IN_BOUNDS, EVT_ITEM_OUT_OF_BOUNDS, EVT_KEY, EVT_OFFSCREEN, EVT_ONSCREEN, EVT_POPUP_CLOSED, EVT_POPUP_PANEL_CLOSED, and EVT_TOUCH, and the following new widgets: GestureArea, KeyboardArea, ScrollWheelArea, MultiStateButton, and TouchArea.

Expanded Table 3.1.4-2 Widget Event Cross Reference with the following events: A661_EVT_OFFSCREEN, A661_EVT_ONSCREEN, A661_EVT_POPUP_CLOSED, A661_EVT_POPUP_PANEL_CLOSED, and A661_EVT_TOUCH and the following new widgets: GestureArea, KeyboardArea, ScrollWheelArea, MultiStateButton, and TouchArea.

3.2.1 Widget Summary

Updated Table 3.2.1 Widget Library Summary to include the following new widgets added for Supplement 6: AnimationGroup, AnimationOnParam, AnimationRotation, AnimationScale, AnimationTranslation, EventHandler, FramingRefreshContainer, GestureArea, InkArea, KeyboardArea, MapBoundary, MapHorz_VertexBuffer, MapVert_Container, MapVert_Panel, MultiStateButton, ScrollWheelArea, and TouchArea added as new widgets for Supplement 6.

3.2.2 Widget Classification

Updated Table 3.2.2-2 Widget Classification Table to include MultiStateButton, KeyboardArea, ScrollWheelArea, MapHorz_VertexBuffer, TouchArea, GestureArea, InkArea, AnimationTranslation, AnimationScale, AnimationGroup, AnimationRotation, AnimationOnParam, MapVert_Container, MapVert_Panel, MapBoundary, DataConnector, EventHandler, and FramingRefreshContainer as new widgets added for Supplement 6. Removed code A UA Validation from the MapHorz_Container widget.

3.2.3.1 Possible Children of Container Widgets

Updated Table 3.2.3.1 Possible Children of Container Widgets to include AnimationGroup, FramingRefreshContainer, MapVert_Container and MapVert_Panel as parents and MultiStateButton, InteractiveKeyboardArea, InteractiveScrollWheelArea, MapHorz_VertexBuffer, TouchInteractionArea, GestureArea, InkArea, AnimationTranslation, AnimationScale, AnimationGroup, AnimationRotation, AnimationOnParam, MapVert_Container, MapVert_Panel, MapBoundary, DataConnector, EventHandler, and FramingRefreshContainer as new widgets added for Supplement 6.

3.2.5.1 Available Character Set and Character Set Encodings

Revised and enhanced descriptions in this section defining the characters and shapes available in this specification. Added equivalent GBK Code Points to Table 3.2.5.1-2 A661 Character Set Extensions.

3.2.5.5 Escape Sequences Description

Relocated and revised the former encoding and escape sequence detail footnotes to main body text.

3.2.11 Animations

Added this new section outlining animations.

3.3 Widget List

Added Param7 in Table 3.3-2 Example of Creation structure. Relocated Table 3.3-3 Notations used in ARINC 661 Command Structures to this section from its previous location in Section 4.5.1.

3.3.22.1 MapHorz_ItemList Standard Items Description

Added BOUNDARY_CHECK, PARKING_LINE_START, PARKING_LINE_END, PARKABLE_SYMBOL, PARKABLE_SYMBOL_ INTERACTIVE VERTEX_RENDER_START, VERTEX_RENDER_SEGMENT, and VERTEX_RENDER_END to Table 3.3.22.1 MapHorz_ItemList Standard Items Description as standard items.

3.3.22.2.1.26 Vertex Render Start

Added this new section to enable rendering of the vertices contained in a MapHorz_VertexBuffer as a sequence of triangles, a triangle fan, triangle strips or a polyline.

3.3.22.2.1.27 Vertex Render Segment

Added this new section which adds a segment to a vertex render start map.

3.3.22.2.1.28 Vertex Render End

Added this new section which terminates vertex render map items.

3.3.22.2.1.29 Parkable Symbol and Parkable_Symbol_Interactive

Added this new section which permit defining a symbol shown on the map based on a map boundary.

3.3.22.1.30 Parking_Line_Start

Added this new section which allows the definition of the start of a line shown on the map based on a Map Boundary.

3.3.22.2.1.31 Parking_Line_End

Added this new section allows the definition of the end of a line shown on the map based on a Map Boundary.

3.3.22.2.1.32 Boundary_Check

Added this new section which indicates whether boundary checking should be performed.

3.3.24 MapHorz_Source

Added description for NbOfIncrements in Table 3.3.24-4a MapHorz_Source Event Structures.

3.3.25 MapHorz

Added runtime modifiability to the PosX, PosY, SizeX, and SizeY parameters in Table 3.3.25-1 MapHorz Parameters. Added PosX, PosY, PosXPosY, SizeX, SizeY and SizeXSizeY parameters in Table 3.3.25-1 MapHorz Runtime Modifiable Parameters

3.3.26 MaskContainer

Enhanced description of the MaskContainer widget.

3.3.37 ScrollList

Included additional clarifying language.

3.4.3 MapVert

Added runtime modifiability to the PosX, PosY, SizeX, and SizeY parameters in Table 3.4.3-1 MapVert Parameters. Added PosX, PosY, PosXPosY, SizeX, SizeY

and SizeXSizeY parameters in Table 3.3.3-3 MapVert Runtime Modifiable Parameters.

3.4.4 MapVert_Source

Corrected value/range for the WidgetType parameter buffer in Table 3.4.4-2a MapVert_Source Creation Structure.

3.4.6 EditBoxMultiLine

Corrected name of Table 3.4.6-5 EditBoxMultiline Event Structure: A661_EVT_STRING_CONFIRMED and value/description of the EventIdent EventStructure.

3.6.4.3 FocusOut

Included additional language in the description for the NextFocusInAppId parameter.

3.8.2 PagingContainer

The description of the PagingContainer widget has been updated and VisibleChildIndex parameter replaced by VisibleChild (visible child is selected by its ident, not its index, as the VisibleChildIndexExtension can be used for index access). Added ReportVisibleChild parameter to Table 3.8.2-1 PagingContainer Parameters. To maintain consistency with the MutuallyExclusiveContainer widget and the VisibleChildIndexExtension extension, the VisibleChild runtime message identifier was renamed A661_VISIBLE_CHILD and the event identifier A661_EVT_VISIBLE_CHILD retained. Added VisibleChild name to the CreateParameterBuffer in Table 3.8.2-2 PagingContainer Creation Structure. Added Widget Ident to the value/description of the VisibleChild EventStructure in Table 3.8.2-3. Added VisibleChild to the name of the parameter to set in Table 3.8.2-4 PagingContainer Runtime Modifiable Parameters.

3.9 Widgets Added for Supplement 6

This section added to provide detailed descriptions of new widgets added for Supplement 6.

4.2.1 Definition File and UALD

Included additional explanatory language.

4.2.2 Binary Format

Replaced Figure 4.2.2 – Definition File Integration Process with Figure 4.2.2a – UA DF Generation and Ownership and Figure 4.2.2b – Example CDS Definition File Relationship.

4.4.3.2 Notification from CDS to UA

Added A661_NOTE_CONNECTOR_DATA as a new notification type.

4.5.1 Notation

This section relocated to Section 3.3 Widget List and named Table 3.3-3 Notations used in ARINC 661 Command Structures.

4.5.3.2 Definition File (DF) Structure

Enhanced description of [A661_Symbol_Block_Structure_DT] in Table 4.5.3.2-1. Added a new definition file [A661_Animation_Law_Block_Structure_DT] to Table 4.5.3.2-1 – Definition File Structure. Revised description of Applildnet in Table 4.5.3.2-2 – Definition File Header. Revised language describing possible values for the A661_Charset_Encoding.

4.5.4.4 Notification Structure

Added Table 4.5.4.4-6 – Connector_Data_Structure.

4.4.6.22 A661_ParameterStructure_BufferOfExclusionItems

Added this new section with references to other sections of the document for additional description.

4.5.4.6.23 A661_ParameterStructure_App_Data

Added this new section defining A661_ParameterStructure_App_Data.

4.5.4.6.24 A661_ ParameterStructure BoundaryArray

Added this new section with references to other sections of the document for additional description.

4.6 ARINC 661 Keyword Values

Added block codes for two new specification ranges A661_BEGIN_ANIMATION_LAW_BLOCK and A661_END_ANIMATION_LAW_BLOCK in Table 4.6-2 – Block Codes.

Added command to the specification range A661_CMD_CREATE_ANIMATION_LAW in Table 4.6-3a – Commands.

Table 4.6-3b Symbol Definition Commands alphabetized and expanded to include A661_SYMBOL_DEFN_SET_SCALE and A661_ DEFN_SET_SYMBOL_REF.

Added Table 4.6-3c – Animation Law Definition Commands.

Alphabetized Table 4.6-5a Notes and added specification range for A661_NOTE_CONNECTOR_DATA.

Table 4.6-7 Widgets expanded to include: A661_ANIMATION_GROUP, A661_ANIMATION_ONPARAM, A661_ANIMATION_ROTATION, A661_ANIMATION_SCALE, A661_ANIMATION_TRANSLATION, A661_DATA_CONNECTOR, A661_EVENT_HANDLER, A661_FRAMING_REFRESH_CONTAINER, GESTUREAREA, INKAREA, KEYBOARDAREA, SCROLLWHEELAREA, A661_MAP_BOUNDARY, MapHorz_VertexBuffer, A661_MAPVERT_CONTAINER, A661_MAPVERT_PANEL, MultiStateButton, and TOUCHAREA as new widgets added for Supplement 6. Deleted the reserved widget keyword value 0xA2E0 which erroneously duplicated the A661_SCALING_FR180 keyword value.

Added A661_ANIMATION_DELAY, A661_ANIMATION_DURATION, A661_ANIMATION_LAW_REF, A661_ANIMATION_REPETITION, A661_ANIMATION_STATE, A661_APP_DATA, A661_BOUNDARY_ARRAY, A661_BOUNDARY_CHECK_MODE, A661_CACHING_ENABLE, A661_CLEAR, A661_EFFECTIVITY_ARRAY, A661_ENABLE_HANDLER, A661_EXCLUDED_REGIONS_LIST, A661_FROM_ANGLE, A661_FROM_SCALE_X, A661_FROM_SCALE_Y, A661_FROM_SCALE_XY, A661_FROM_VALUE, A661_FROM_VALUE_X, A661_FROM_VALUE_Y, A661_FROM_VALUE_XY, A661_KEY_ARRAY, A661_MULTI_STATE_VALUE, A661_NUMBER_OF_KEYS, A661_NUMBER_OF_TICS_COARSE, A661_NUMBER_OF_TICS_FINE, A661_NUMBER_OF_TOUCH_POINTS, A661_NUMBER_OF_STATES, A661_PARAM_BUFFER_INDEX, A661_SELECT_ALL, A661_SELECTED_ENTRY_ARRAY, A661_STYLE_SET_ARRAY, A661_STYLE_SET_ARRAY_SIZE, A661_SYMBOL_ENTRY_ARRAY, A661_SYMBOL_ENTRY_ARRAY, A661_SYNC_FRAME, A661_TICS_COARSE_ARRAY, and

A661_TICS_FINE_ARRAY, A661_TO_ANGLE, A661_TO_SCALE_X, A661_TO_SCALE_Y, A661_TO_SCALE_XY, A661_TO_VALUE, A661_TO_VALUE_X, A661_TO_VALUE_Y, A661_TO_VALUE_XY, A661_DIRECTION, A661_NUMBER_OF_TURNS, A661_UNSELECT_ALL as parameter types in Table 4.6-8 Parameter Types.

Added A661_EVT_ANIMATION_STATUS_CHANGE, A661_EVT_GESTURE_DIR_FLICK, A661_EVT_GESTURE_DRAG, A661_EVT_GESTURE_FLICK, A661_EVT_GESTURE_DOUBLE_TAP, A661_EVT_GESTURE_TAP, A661_EVT_ITEM_IN_BOUNDS, A661_EVT_ITEM_OUT_OF_BOUNDS, A661_EVT_KEY, A661_EVT_MULT_SELECTION, A661_EVT_GESTURE_PINCH, A661_EVT_GESTURE_PRESS_AND_HOLD, A661_EVT_GESTURE_ROTATE, A661_EVT_TOUCH, and A661_EVT_VISIBLE_CHILD_INDEX as events types in Table 4.6-9 Event Types.

Assigned integer constant values for A661_ABORT, A661_ABORTED, A661_ARC, A661_BOUNDARY_CHECK, A661_CLOCKWISE, A661_COMPLETED, A661_COUNTER_CLOCKWISE, A661_EXCLUDED_RECTANGLE, A661_EXCLUDED_TRIANGLE, A661_EXCLUDED_CIRCLE, A661_GESTURE_DOUBLE_TAP, A661_GESTURE_DRAG, A661_GESTURE_END, A661_GESTURE_FLICK, A661_GESTURE_FLICK_DOWN, A661_GESTURE_FLICK_LEFT, A661_GESTURE_FLICK_RIGHT, A661_GESTURE_FLICK_UP, A661_GESTURE_PINCH, A661_GESTURE_PRESS_AND_HOLD, A661_GESTURE_ROTATE, A661_GESTURE_START, A661_GESTURE_TAP, A661_GESTURE_UPDATE, A661_HORZ_LINE, A661_IN_BOUNDS, A661_MINIMUM_SIZE, A661_OPERATOR_ADD_ASSIGN_ARG, A661_OPERATOR_ADD_ASSIGN_SRC, A661_OPERATOR_ASSIGNMENT, A661_OPERATOR_CONSTANT, A661_OPERATOR_INVERSE, A661_OPERATOR_LOGICAL_AND, A661_OPERATOR_LOGICAL_OR, A661_OPERATOR_PRODUCT, A661_OPERATOR_PROD_ASSIGN_ARG, A661_OPERATOR_PROD_ASSIGN_SRC, A661_OPERATOR_SUB_ASSIGN_ARG, A661_OPERATOR_SUB_ASSIGN_SRC, A661_OPERATOR_SUM, A661_OUT_OF_BOUNDS, A661_PARALLEL, A661_PARKING_LINE_START, A661_PARKING_LINE_END, A661_PLAY, A661_REPORT_ON_COMPLETION, A661_SEQUENTIAL, A661_SOURCE_WIDGET_PARAM, A661_SYMBOL_PARKABLE, A661_SYMBOL_PARKABLE_INTERACTIVE, A661_TOUCH_DOWN, A661_TOUCH_MOVE, A661_TOUCH_UP, A661_VERT_LINE, A661_REP_DEFAULT, A661_REP_STANDARD, A661_REP_FOCUS, A661_REP_HIGHLIGHT, A661_VERTEX_RENDER_START, A661_VERTEX_RENDER_SEGMENT, A661_VERTEX_RENDER_END, A661_VERTEX_RENDER_LINES, A661_VERTEX_RENDER_LINE_LOOP, A661_VERTEX_RENDER_POLYLINE, A661_VERTEX_RENDER_TRIANGLE, A661_VERTEX_RENDER_TRIANGLE_FAN, and A661_VERTEX_RENDER_TRIANGLE_STRIP in Table 4.6-11 Integer Constant Values.

Added A661_BOUNDARY_CHECK_EXTENSION, A661_PICTURE_EXTENSION, A661_STATIC_PARAM_BUFFER_EXTENSION, A661_STYLE_SET_EXTENSION, A661_SYMBOL_EXTENSION, and A661_TICS_ARRAY_EXTENSION as new widget extensions in Table 4.6-12 Widget Extensions.

Added Table 4.6-14 – Animation Laws.

5.2.3.3 Scale

Added this new section describing the SET_SCALE command.

5.2.4.16 Set SymbolReference

Added this new section describing the SET_SYMBOL_REF command.

6.1.1 XML Definition File XSD

Added this new section defining the interface syntax. The schema and associated examples are now available at the following URL: http://www.aviation-ia.com/aeec/SupportFiles/661-6/df.zip.

6.2. Description

Deleted Table 6.2-1 – XML Element Table Format, Table 6.2-2 – a661_df Element and verbal descriptions which are now available in the electronic support files.

6.2.1 Charset Encoding

Revised wording of this section and deleted Table 6.2.1-1 – charset-encoding XML Element.

6.2.1.1 ASCII Extended Character Mappings

Added this new section delineating the extended ASCII character mappings.

6.2.2 Symbol Graphical Definitions, Picture Definitions and Animation Law Definitions.

Renamed this section from the previous Picture and Symbol Graphical Definitions, revised wording, and deleted Table 6.2.2-1 – symboltable XML Element, Table 6.2.2-2 – symboldefn XML Element, Table 6.2.2-3 – stdrepr XML Element, Table 6.2.2-3 – stdrepr XML Element, Table 6.2.2-4 – focusrepr XML Element, Table 6.2.2-5 – highlightrepr XML Element, Table 6.2.2-6 – rectangular_sensitive_area XML Element, Table 6.2.2-7 – circular_sensitive_area XML Element, Table 6.2.2-8 – symboldefncmd XML Element, Table 6.2.2-9 – picturetable XML Element, Table 6.2.2-10 – picturedefn XML Element which are now available in the electronic support files.

6.2.3 Layers and Widgets

This section including Table 6.2.3-1 – a661_layer XML Element, Table 6.2.3-2 – a661_widget XML Element, and Table 6.2.3-3 – a661_extension XML Element deleted and now available in the electronic support files.

6.2.4 Properties

Revised wording in this section.

6.2.4.1 General Property Considerations

Added this new section with contents from the former Section 6.2.4 Properties and included enhanced language regarding UTF-8 XML encoding and binary representation.

6.2.4.2 Backslash-Escaping

Added this new section with contents from the former Section 6.2.4 Properties and new material delineating backslash-escaping. Deleted Table 6.2.4-1 – model XML Element, Table 6.2.4-2 – prop XML Element, Table 6.2.4-3 – structprop XML Element, Table 6.2.4-4 – arrayprop XML Element, Table 6.2.4-5 – field XML Element, Table 6.2.4-6 – structfield XML Element, Table 6.2.4-7 – arrayfield XML Element, Table 6.2.4-8 – entry XML Element, Table 6.2.4-9 – xyentry XML Element, Table 6.2.4-10 – structentry XML Element, and Table 6.2.4-11 – arrayentry XML Element. These tables are now available in the electronic support files.

6.2.4.3 Variable-Structure Properties

Added this new section delineating the datablock, datum and datapair elements utilized for variable-structure properties.

6.3 Document Type Definition (DTD) Specification

This section deleted in Supplement 6.

8.3 Possible Widgets for Extension

Added PictureExtension, SymbolExtension, TicsArrayExtension, StyleSetExtension, StaticParamBufferExtension, MultiSelectionExtension, ExcludedRegionsExtension, and BoundaryCheckExtension as new extensions in Table 8.3 Possible Widgets for Extension. Also added AnimationGroup, AnimationOnParam, AnimationRotation, AnimationScale, AnimationTranslation, DataConnector, EventHandler, FramingRefreshContainer, GestureArea, InkArea, KeyboardArea, MapBoundary, MapHorz_VertexBuffer, MapVert_Container, MapVert_Panel, MultiStateButton, ScrollWheelArea, and TouchArea as new widgets added for Supplement 6.

8.4 Widget Extensions Added for Supplement 6

Renamed this section from the former Extension List.

8.4.1 DirectionalTabbingExtension

Removed ActiveArea, CheckButton, ComboBox, EditBoxMasked, EditBoxNumeric, EditBoxText, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ScrollList, TabbedPanel, ToggleButton, EditBoxMultiLine, ComboBoxEdit, Slider, SelectionListButton, EditBoxNumericBCD, SymbolPushButton, and SymbolToggleButton as possible widgets for DirectionalTabbingExtension as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.4.2 CursorEventsExtension

Removed ActiveArea, PicturePushButton, PushButton, and SymbolPushButton, as possible widgets for **CursorEventsExtension** as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.4.3 LegendStringAlignmentExtension

Removed EditBoxNumeric, EditBoxNumericBCD, and NumericReadout as possible widgets for **LegendStringAlignmentExtension** as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.4.4 VisibleChildIndexExtension

Enhanced descriptive notes and updated to indicate that the first child of a given parent corresponds to index 1. Clarified that the description of the VisibleChildIndex event shall be sent only when the extended widget allows events to be reported (i.e., anytime an active child changes for TabbedPanelGroup widget, or only when ReportVisibleChild definition time parameter is set to true and an active child changes for PagingContainer). Removed MutuallyExclusiveContainer, TabbedPanelGroup, and PagingContainer, as possible widgets for VisibleChildIndexExtension as they are delineated in Table 8.3 – Possible Widgets for Extension. Updated the VisibleChildIndex description in Table 8.4.4-1 VisibleChildIndexExtension Parameters to indicate that index 0 means no visible child.

8.4.5 InitialFocusExtension

Removed ActiveArea, CheckButton, ComboBox, EditBoxMasked, EditBoxNumeric, EditBoxText, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ScrollList, TabbedPanel, ToggleButton, EditBoxMultiLine, ComboBoxEdit, Slider, SelectionListButton, EditBoxNumericBCD, SymbolPushButton, SymbolToggleButton and PopUpPanelButton as possible widgets for DirectionalTabbingExtension as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.4.6 FocusStopExtension

Removed ActiveArea, CheckButton, ComboBox, EditBoxMasked, EditBoxNumeric, EditBoxText, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ScrollList, TabbedPanel, ToggleButton, EditBoxMultiLine, ComboBoxEdit, Slider, SelectionListButton, EditBoxNumericBCD, SymbolPushButton, SymbolToggleButton and PopUpPanelButton as possible widgets for FocusStopExtension as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.4.7 CursorShapeExtension

Removed ActiveArea, CheckButton, ComboBox, EditBoxMasked, EditBoxNumeric, EditBoxText, PicturePushButton, PictureToggleButton, PopUpMenuButton, PushButton, ScrollList, TabbedPanel, ToggleButton, EditBoxMultiLine, ComboBoxEdit, Slider, SelectionListButton, EditBoxNumericBCD, SymbolPushButton, SymbolToggleButton and PopUpPanelButton as possible widgets for CursorShapeExtension as they are delineated in Table 8.3 – Possible Widgets for Extension.

8.5.1 PictureExtension

Added this section describing the PictureExtension which allows the addition of one or more pictures to a widget.

8.5.2 SymbolExtension

Added this section describing the SymbolExtension which allows the addition of one or more symbols to a widget.

8.5.3 TicsArrayExtension

Added this section describing the TicsArrayExtension which enables variable calibration of a widget.

8.5.4 StyleSetExtension

Added this new section describing the StyleSetExtension which allows an additional StyleSet capacity for implementations which require more than 16 bits.

8.5.5 StaticParamBufferExtension

Added this new section describing the StaticParamBufferExtension which stores a static array of BufferFormatData values within the CDS.

8.5.6 MultiSelectionExtension

Added this new section describing the MultiSelectionExtension which allows the operator to select multiple entries for a widget.

8.5.7 ExcludedRegionsExtension

Added this new section describing the ExcludedRegionsExtension which defines a list of regions that exclude parkable map items.

8.5.8 BoundaryCheckExtension

Added this new section describing the BoundaryCheckExtension which provides a mechanism for the UA to indicate how boundary check results should be reported.

9.0 Animation Laws Definition

Added this new section describing how animation laws can be defined. These laws are used to provide animation capabilities.

APPENDIX B – ACRONYMS AND ABBREVIATIONS

XSD was added.

APPENDIX C – EXAMPLE OF THE XML FORM OF THE DEFINITION FILE

C-5 A More Interesting XML DF Example

Enhanced description, corrected XML file example.

C-12 XML DF Specifying Escape Sequences

Enhanced description.

C-13 XML DF Specifying Variable-Structure Properties

Added this new section providing an example of how to store a StaticParamBufferExtension's BufferArray in XML format.

APPENDIX G – NEW WIDGET GUIDELINES

G-7 Creation Structure

Added DataConnector as an exception to the Enable flag going just before the Visible Flag rule.

APPENDIX H – CDS LAYER AND WINDOW MANAGEMENT

H-3 Schema Mapping to Widgets

Changed DataConnector reference from Section H-6 to Section 3.9.16.

H-5.1 Window Manager to UA Communication

Changed DataConnector reference from Section H-6 to Section 3.9.16.

H-6.2 DataConnector

Relocated this section to section 3.9.16 and enhanced.

APPENDIX I – ARINC 661 UA DESIGN CONSIDERATIONS

I-2 Main Concepts and High-Level View of ARINC 661 UA-Software Architecture

Revised wording and updated Figure I-2-1 – ARINC 661 Abstractions.

I-4.1 Simple Library

Updated Figure I-4.1-1 – Simple Library UA Software Architecture.

I-4.2 Basic API

Updated Figure I-4.2-1 – Basic API UA Software Architecture.

I-4.3 API with Framework

Updated Figure I-4.3-1 – API with Framework US Software Architecture.

I-4.4.2 Probable components for this approach

Updated Figure I-4.4.2-1 – Full Abstraction (with Framework).

I-5 Proxy User Applications

Added this new section outlining proxy user applications.

APPENDIX J – LOOK MODELING

J-4.1.1 Look Capacities and Look Definition XSD (XML Schema Document)

Relocated the contents of the former J-4.2 Look Capacities and Look Definition XSD (XML Schema Document) here.

Deleted Section J-4.1.1 Look General XML Elements Description and Table J-4.1.1-1 – Look XML Element, Table J-4.1.1-2 – Constants XML Element, Table J-4.1.1-3 – Constant XML Element, Table J-4.1.1-4 – ConstantProperty XML Element, Table J-4.1.1-5 – FreeText XML Element, Table J-4.1.1-6 – Value XML Element, and Table J-4.1.1-7 – ValueRef XML Element as these tables are now available on the electronic support files.

Deleted Section J-4.1.2 Look Capacities XML elements description and Table J-4.1.2-1 – LookCapacities XML Element, Table J-4.1.2-2 – WidgetsCapacities XML Element, Table J-4.1.2-3 – WidgetCapacities XML Element, Table J-4.1.2-4 – ComponentsCapacities XML Element, Table J-4.1.2-5 – ComponentCapacities XML Element, Table J-4.1.2-6 – Attributes XML Element, Table J-4.1.2-7 – Attribute XML Element, Table J-4.1.2-8 – States XML Element, Table J-4.1.2-9 – State XML Element, Table J-4.1.2-10 – Types XML Element, Table J-4.1.2-11 – Type XML Element, Table J-4.1.2-12 – Parameter XML Element, Table J-4.1.2-13 – Enumerates XML Element, Table J-4.1.2-14 – Enumerate XML Element, and Table J-4.1.2-15 – Literal XML Element as these tables are now available on the electronic support files.

Deleted Section J-4.1.3 Look Definition XML Elements Description and Table J-4.1.3-1 – LookDefinition XML Element, Table J-4.1.3-2 – WidgetsDefinitions XML Element, Table J-4.1.3-3 – WidgetDefinition XML Element, Table J-4.1.3-4 – ComponentsDefinitions XML Element, Table J-4.1.3-5 – ComponentDefinition XML Element, Table J-4.1.3-6 – Styleset XML Element, Table J-4.1.3-7 – Properties XML Element, Table J-4.1.3-8 – Property XML Element, Table J-4.1.3-9 – StateDefinition XML Element, Table J-4.1.3-10 – StatesList XML Element, Table J-4.1.3-11 – StateItem XML, Element, Table J-4.1.3-12 – Tables XML Element, Table J-4.1.3-13 – Table XML Element, Table J-4.1.3-14 – Entry XML Element, and Table J-4.1.3-15 – EntryProperty XML Element as these tables are now available on the electronic support files.

J-4.1.2 Naming Rules Recommendations

Renumbered this section from the former J-4.1.4.

J-4.1.3 Reserved Words XML Elements Description

Renumbered this section from the former J-4.1.5.

J-4.1.4 Optional Default Values for the Attributes in the Look Capacities

This new section added describing how to specify the default value of an attribute.

J-4.1.5 Optional Bitmask Position and Size for the Attributes in the Look Capacities

This section added describing how to specify the bit mask where each attribute element may be found.

J-4.1.6 Relationship with the Styleset Extension

This section added describing how StyleSets are defined.

J-4.2 Look Capacities and Look Definition XSD (XML Document)

This section deprecated in Supplement 6.

J-4.3 Look Capacities and Look Definition XML examples

Contents of this section deleted in Supplement 6, as they are now contained in the electronic support files.

APPENDIX K – UA/CDS INTERFACE

K-5.1 UA/CDS Interface XSD (XML Schema Document)

Renumbered and relocated the former Section K-5.2 here, updated the URL and deleted Table K-5.1-1 – FreeText XML Element, Table K-5.1-2 – UaCdsInterface XML Element, Table K-5.1-3 – UaCdsInterfaceCapacities XML Element, Table K-5.1-4 – UaCdsInterfaceDefinition XML Element, Table K-5.1-5 – a661BufferFillingModalityTypes XML Element, Table K-5.1-6 – a661BufferFillingModalityType XML Element, Table K-5.1-7 – Property XML Element, Table K-5.1-8 – a661BufferFillingModalities XML Element, Table K-5.1-9 – a661BufferFillingModality XML Element, Table K-5.1-10 – a661BufferFillingModalityProperty XML Element, Table K-5.1-11 – Interfaces XML Element, Table K-5.1-12 – InterfacesGroups XML Element, Table K-5.1-13 – InterfacesGroup XML Element, Table K-5.1-14 – CommandInterfaces XML Element, Table K-5.1-15 – CommandInterface XML Element, Table K-5.1-16 – NotificationInterfaces XML Element, Table K-5.1-17 – NotificationInterface XML Element, Table K-5.1-18 – GraphicalGroups XML Element, Table K-5.1-19 – GraphicalGroup XML Element, Table K-5.1-20 – Commands XML Element, Table K-5.1-21 – Command XML Element, Table K-5.1-22 – Request XML Element, Table K-5.1-23 – SetParameter XML Element, Table K-5.1-24 – Notifications XML Element, Table K-5.1-25 – Notification XML Element, Table K-5.1-26 – LayerEvent XML Element, Table K-5.1-27 – layerError XML Element, Table K-5.1-28 – WidgetEvent XML Element, Table K-5.1-29 – WidgetError XML Element, and Table K-5.1-30 – NotificationData XML Element as these tables are now available on the electronic support files.

K-5.2 Extension Mechanism

Renumbered from the previous Section K-5.2.

K-5.4 UA/CDS Interface XML Example

Renumbered from the previous Section K-5.3 and deleted the example which is now available in the electronic support files.

APPENDIX L – ASCII EXTENDED CHARACTER SET MAPPINGS

Added this new section defining an XML format which specifies how ASCII Extended Characters map to UNICODE code points.

SAE INDUSTRY TECHNOLOGIES CONSORTIA (SAE ITC)
16701 Melford Blvd., Suite 120
Bowie, Maryland 20715 USA

SUPPLEMENT 7 TO

ARINC SPECIFICATION 661

COCKPIT DISPLAY SYSTEM INTERFACE TO USER SYSTEMS
PART 1
AVIONICS INTERFACES, BASIC SYMBOLOGY, AND BEHAVIOR

Published: June 17, 2019

Prepared by the AEEC

Adopted by the AEEC Executive Committee: May 1, 2019

A. PURPOSE OF THIS DOCUMENT

Supplement 7 introduces numerous changes and additions to **ARINC Specification 661:** *Cockpit Display System Interface to User Systems, Part 1, Avionics Interfaces, Basic Symbology, and Behavior*.

B. ORGANIZATION OF THIS SUPPLEMENT

In this document **blue bold** text is used to indicate those areas of text changed by the current supplement only.

C. CHANGES TO ARINC SPECIFICATION 661 PART 1 INTRODUCED BY THIS SUPPLEMENT

This section presents a complete listing of the changes to the document introduced by this Supplement. Each change is identified by the section number and the title as it will appear in the complete document. Where necessary, a brief description of the change is included.

1.2 Relationship to Other Documents

Added **ARINC Characteristic 739A:** *Multi-Purpose Control and Display Unit*.

2.3.2.3 Layer Priority Management

Added clarifying language for PopUp widgets.

2.3.4.1 Origins

In Figure 2.3.4, added clarifying language for directional pointing of x, y, and z axes.

3.1.2.1 Widget States Definition

Added “Interactivity” to describe the level.

3.1.3.2 States of a Widget

In Table 3.1.3.2, Widget State Parameters, for “Enable”, added clarifying language to the Description.

3.1.3.3 Look and Feel Characteristics of a Widget/Symbol

In Table 3.1.3.3, Widget StyleSet Parameter, added “StartCursor-Pos/StartCursor-PosByte” Parameter and Description.

3.1.3.3.1 Predefined Look-Up Look and Feel Characteristics

This section was added to define characteristics that provide a common means to describe control look and feel attributes.

3.1.3.4 Positioning/Size of a Widget

Added language clarifying clipping implementation for graphical characteristics outside the area.

3.1.4 Widget Events

In Table 3.1.4-1, Widget Event Cross Reference, added Event “A661_EVT_GESTURE_HOLD” and added missing “EVT” to Event labels.

In Table 3.1.4-2, Widget Event Cross Reference, added Widget “Selector”.

3.2.1 Widget Summary

In Table 3.2.1, Widget Library Summary, added Widget Types:

Widgets Added for Supplement 6: “DataConnector”

Widgets Added for Supplement 7: “GpTriangleFan”, “GpTriangleStrip”, “MapExternalSource”, “ScaleContainer”, “Selector”, and “Tree”

3.2.2 Widget Classification

In Table 3.2.2-1, Widget Library Categories, added Description clarification for Widget Category “UA Validation” and deleted “Dynamic Motion”.

In Table 3.2.2-2, Widget Classification Table, removed “Dynamic Motion” column, added Widgets Added for Supplement 7: “ScaleContainer”, “Selector”, “Tree”, “MapExternalSource”, GpTriangleFan”, and “GpTriangleStrip”, and added “A” for UA Validation for “KeyboardArea” and “ScrollWheelArea”.

3.2.3.1 Possible Children of Container Widgets

In Table 3.2.3.1, Possible Children of Container Widgets:

Added “ScaleContainer” to Parents

“ScaleContainer” was made Parent of the following Children: “BlinkingContainer”, “Connector”, “GpArcCircle”, “GpArcEllipse”, “GpCrown”, GpLine”, “GpLinePolar”, “GpRectangle”, “GpTriangle”, “Label”, “Picture”, “RotationContainer”, “Symbol”, ”TranslationContainer”, “External Source”, “PictureAnimated”, “SymbolAnimated”, ”GpPolyline”, “NumericReadout”, ”DataConnector”, “GpTriangleFan”, “GpTriangleStrip”, and “ScaleContainer”

Added Widgets Added for Supplement 7: “GpTriangleFan”, “GpTriangleStrip”, “ScaleContainer” ”, “Selector”, and “Tree” as Children with various Parents

“MaskContainer” was made Parent of “MapHorz” and “MapVert”

“MapVert” was made Parent of “MaskContainer”

“RotationContainer” and “TranslationContainer” were made Parent of “Picture” and “PictureAnimated”

“ShuffleToFitContainer” was made Parent of “CursorRef”

“SizeToFitContainer” and “ShuffleToFitContainer” was made Parent of “EditBoxNumericBCD”

“MapExternalSource” was deleted

“SymbolPushButton” was removed as a Child of Parent “”RadioBox”

3.2.5 Text Strings

Added clarifying language regarding character encoding.

3.2.5.1 Available Character Set and Character Set Encodings

In Table 3.2.5.1-2, A661 Character Set Extensions, added the following characters: “Figure Space”, “Filled up pointing triangle”, “Filled down pointing triangle”, “Filled right pointing triangle”, “Filled left pointing triangle”, and “Horizontal ellipsis”.

3.2.6 Formatted Numeric Values

This section was added to define how widgets and MapItems may be represented as a numeric value, according to a format string.

3.2.8 Dynamic Motion

This widget deprecated by Supplement 7.

3.2.9.5 MapItem Legends

This section was added to describe MapItems (called LEGENDxxx) which use Legend Strings to present Legends or format numeric values in MapHorzItemList or MapVertItemList widgets.

3.2.12 Animations

The animation state transition algorithm was updated to describe the actions when the AnimationState parameters are set to one of the following: A661_PLAY, A661_ABORT, A661_ABORTED, A661_DEACTIVATE, A661_END, and A661_COMPLETED, A661_FROM, and A661_TO.

A Notes section was added to the animation state transition algorithm section.

Three examples were added:

Two children animations on two different widgets

Two children animations on the same widget

Two transformation animations on the same widget

3.3.1 ActiveArea

In Table 3.3.1-1, ActiveArea Parameters, “PosX”, “PosY”, “SizeX”, and “SizeY” Change value was changed from “D” to “DR”.

In Table 3.3.1-4, ActiveArea Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, and “SizeX, SizeY”.

3.3.2 BasicContainer

Category “Dynamic Motion” was deleted.

Sentence in the Commentary, “BasicContainer can be used to define, at run-time, the position of a button.” was deleted.

3.3.5 CheckButton

In Table 3.3.5-1, CheckButton Parameters:

“MaxStringLength” Description was changed

“PosX”, “PosY”, “SizeX”, “SizeY”, “Alignment”, and “PicturePosition” Change value was changed from “D” to “DR”

In Table 3.3.5-4, CheckButton Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, and “PicturePosition”.

3.3.6 ComboBox

In Table 3.3.6-1, ComboBox Parameters:

“MaxStringLength” Description was changed

“PosX”, “PosY”, “SizeX”, “SizeY”, “SelectingAreaWidth”, “SelectingAreaHeight”, “OpeningMode”, and “Alignment” Change value “D” was changed to “DR”

In Table 3.3.6-4, ComboBox Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “SelectingAreaWidth”, “SelectingAreaHeight”, “OpeningMode”, and “Alignment”.

3.3.8 CursorPosOverlay

In Table 3.3.8-1, CursorPosOverlay Parameters, “PosX”, “PosY”, “SizeX”, and “SizeY” Change value was changed from “D” to “DR”.

In Table 3.3.8-4, CursorPosOverlay Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, and “SizeX, SizeY”.

3.3.9 EditBoxMasked

In Table 3.3.9-1, EditBoxMasked Parameters, the “StartCursorPos” Description was expanded to add a reference to Section 3.1.3.3.

3.3.10 EditBoxNumeric

In Table 3.3.10-1, EditBoxNumeric Parameters:

- “PosX”, “PosY”, “SizeX”, and “SizeY” Change value was changed from “D” to “DR”
- For the “FormatString” Description, the FormatString form example was deleted and a reference to Section 3.2.6, Formatted Numeric Values, was added
- The “MaxFormatStringLength” Description was clarified
- The “StartCursorPosByte” Description was expanded to add a reference to Section 3.1.3.3
- The “Alignment” Change value was changed from “D” to “DR” and in the Description, “A661_” was added to variables CENTER, LEFT, and RIGHT
- For “NumericKeyFlag”, in the Description, “A661_” was added to variables TRUE and FALSE
- For “LegendAreaSizeX”, Change value was changed from “D” to “DR”.
- The “LegendPosition”, in the Description, “A661_” was added to variables LEFT, RIGHT, TOP, and BOTTOM
- For “LegendRemoved”, Change value was changed from “D” to “DR”
- For the “MinValue” and “MaxValue” the Description was added
- For the “CyclicFlag”, the Description was updated for clarification

In Table 3.3.10-2, EditBoxNumeric Creation Structure, for “FormatString”, the Description was clarified.

In Table 3.3.10-7, EditBoxNumeric Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “LegendAreaSizeX”, and “LegendRemoved”.

3.3.11 EditBoxText

In Table 3.3.11-1, EditBoxText Parameters:

- “PosX”, “PosY”, “SizeX”, “SizeY”, and “Alignment” Change values were changed from “D” to “DR”
- “MaxStringLength” Description was clarified
- “StartCursorPos” Description was expanded to add a reference to Section 3.1.3.3

In Table 3.3.11-7, EditBoxText Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, and “Alignment”.

3.3.12 GpArcEllipse

Category “Dynamic motion” was deleted.

In Table 3.3.12-1, GpArcEllipse Parameters, “ColorIndex” and “FillIndex” Change values were changed from “D” to “DR” and their Descriptions were expanded to add a reference to Section 3.1.3.3.1.

In Table 3.3.12-3, GpArcEllipse Runtime Modifiable Parameters, the following Parameters were added: “Halo” and “Filled”.

3.3.13 GpArcCircle

Category “Dynamic motion” was deleted.

In Table 3.3.13-1, GpArcCircle Parameters:

"ColorIndex" and "FillIndex" Descriptions were expanded to add a reference to Section 3.1.3.3.1

"Halo" and "Filled" Change values were changed from "D" to "DR"

In Table 3.3.13-3, GpArcCircle Runtime Modifiable Parameters, the following Parameters were added: "Halo" and "Filled".

3.3.14 GpCrown

Category "Dynamic motion" was deleted.

In Table 3.3.14-1, GpCrown Parameters:

"ColorIndex" and "FillIndex" Descriptions were expanded to add a reference to Section 3.1.3.3.1.

"Halo" and "Filled" Change values were changed from "D" to "DR".

In Table 3.3.14-3, GpCrown Runtime Modifiable Parameters, the following Parameters were added: "Halo" and "Filled".

3.3.15 GpLine

Category "Dynamic motion" was deleted.

In Table 3.3.15-1, GpLine Parameters:

"ColorIndex" Description was expanded to add a reference to Section 3.1.3.3.1.

"Halo" Change value was changed from "D" to "DR".

In Table 3.3.15-3, GpLine Runtime Modifiable Parameters, the following Parameter was added: "Halo".

3.3.16 GpLinePolar

Category "Dynamic motion" was deleted.

In Table 3.3.16-1, GpLinePolar Parameters:

"ColorIndex" Description was expanded to add a reference to Section 3.1.3.3.1.

"Halo" Change value was changed from "D" to "DR".

In Table 3.3.16-3, GpLinePolar Runtime Modifiable Parameters, the following Parameter was added: "Halo".

3.3.17 GpRectangle

Category "Dynamic motion" was deleted.

In Table 3.3.17-1, GpRectangle Parameters:

"ColorIndex" and "FillIndex" Descriptions were expanded to add a reference to Section 3.1.3.3.1

"Halo" and "Filled" Change values were changed from "D" to "DR"

In Table 3.3.17-3, GpRectangle Runtime Modifiable Parameters, the following Parameters were added: "Halo" and "Filled".

3.3.18 GpTriangle

Category "Dynamic motion" was deleted.

In Table 3.3.18-1, GpTriangle Parameters:

"ColorIndex" and "FillIndex" Descriptions were expanded to add a reference to Section 3.1.3.3.1

"Halo" and "Filled" Change values were changed from "D" to "DR"

In Table 3.3.18-3, GpTriangle Runtime Modifiable Parameters, the following Parameters were added: “Halo” and “Filled”.

3.3.19 Picture

Category “Dynamic motion” was deleted.

In Table 3.3.19-3, Picture Runtime Modifiable Parameters, the following Parameters were added: “PosX, PosY”, “PosX”, “PosY”, “SizeX”, “SizeY”, and “SizeX, SizeY”.

A comment section was added at the end of the section concerning image resolution compared to the picture widget size.

3.3.20 Label

Category “Dynamic motion” was deleted.

In Table 3.3.20-1, Label Parameters:

“MaxStringLength” Description was clarified

“Font” and “ColorIndex” Parameter Descriptions was expanded to add a reference to Section 3.1.3.3.1

“SizeX”, “SizeY”, and “Alignment” Change values were changed from “D” to “DR”

In Table 3.3.20-3, Label Runtime Modifiable Parameters, the following Parameters were added: “SizeX”, “SizeY”, “SizeX, SizeY”, and “Alignment”.

3.3.21 LabelComplex

In Table 3.3.21-1, LabelComplex Parameters:

“MaxStringLength” Description was clarified

“PosX”, “PosY”, “SizeX”, “SizeY”, and “Alignment” Change values were changed from “D” to “DR”

In Table 3.3.21-3, LabelComplex Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, and “Alignment”.

3.3.22 MapHorz_ItemList

For Table 3.3.22-1, MapHorz_ItemList Parameters, Parameter “MaxItemSize” was added.

For Table 3.3.22-2, MapHorz_ItemList Creation Structure, the following Parameters were added: “MaxItemSize” and “UnusedPad”.

3.3.22.1 MapHorz_ItemList Standard Items Description

For Table 3.3.22.1, MapHorz_ItemList Standard Items Description:

The column “Size in Words” was added

Name of Item “LEGEND” Function was clarified

For Name of Item “LEGEND”, “LEGEND_COMBO”, “LEGEND_HIGHLIGHT”, “LEGEND_POP_UP”, “LEGEND_READOUT”, “LEGEND_READOUT_FORMAT_STRING”, and “LEGEND_READOUT_COMBO” the Function was changed to refer to Section 3.2.9.5

Name of Item “PARKABLE_SYMBOL” was changed to “SYMBOL_PARKABLE”

The following Name of Items were added:
“FILLED_POLY_START_INTERACTIVE“, “LEGEND_READOUT”, “LEGEND_READOUT_FORMAT_STRING”, “LEGEND_READOUT_COMBO”, “LINE_START_INTERACTIVE”, “LINE_SEGMENT_INTERACTIVE”, “LINE_ARC_INTERACTIVE”, “SYMBOL_PARKABLE_ROTATED”,

"SYMBOL_PARKABLE_ROTATED_INTERACTIVE",
"SYMBOL_PARKABLE_TARGET",
"SYMBOL_PARKABLE_TARGET_INTERACTIVE",
"SYMBOL_GENERIC_INTERACTIVE",
"SYMBOL_ROTATED_INTERACTIVE",
"SYMBOL_CIRCLE_INTERACTIVE", "SYMBOL_OVAL_INTERACTIVE",
"SYMBOL_RUNWAY_INTERACTIVE",
"SYMBOL_TARGET_INTERACTIVE",
"SYMBOL_RECTANGLE_INTERACTIVE",
"TRIANGLE_STRIP_START_INTERACTIVE",
"TRIANGLE_FAN_START_INTERACTIVE",
"TRIANGLE_SEGMENT_INTERACTIVE",
"TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE",
"TRIANGLE_END_INTERACTIVE", and
"TRIANGLE_END_DOUBLE_INTERACTIVE"

3.3.22.2.1 Item Structures

The Commentary was expanded to add information for maximum size for any MIL item or by using the MaxItemSize parameter.

3.3.22.2.1.3 Legend, Legend_Highlight, and Legend_Pop_Up

In Table 3.3.22.2.1.3, Legend and Legend_Pop_Up, for Name "LegendString," Type and Description and Value/Range were clarified.

3.3.22.2.1.4 Line_Start and Line_Start_Interactive

"Line_Start" is replaced by "Line_Start and Line_Start_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.4.

In Table 3.3.22.2.1.4, Line_Start and Line_Start_Interactive, for Name "ItemType" added "A661_Line_Start_Interactive" in Description and Value/Range.

3.3.22.2.1.5 Line_Segment and Line_Segment_Interactive

"Line_Segment" is replaced by "Line_Segment and Line_Segment_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.5.

In Table 3.3.22.2.1.5, Line_Segment and Line_Segment_Interactive, for Name "ItemType" added "A661_Line_Segment_Interactive" in Description and Value/Range.

3.3.22.2.1.6 Line_Arc and Line_Arc_Interactive

"Line_Arc" is replaced by "Line_Arc and Line_Arc_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.6.

In Table 3.3.22.2.1.6, Line_Arc and Line_Arc_Interactive, for Name "ItemType" added "A661_Line_Arc_Interactive" in Description and Value/Range.

3.3.22.2.1.8 Symbol_Generic and Symbol_Generic_Interactive

"Symbol_Generic" is replaced by "Symbol_Generic and Symbol_Generic_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.8.

In Table 3.3.22.2.1.8, Symbol_Generic and Symbol_Generic_Interactive, for Name "ItemType" added "A661_Symbol_Generic_Interactive" in Description and Value/Range.

3.3.22.2.1.9 Symbol_Circle and Symbol_Circle_Interactive

"Symbol_Circle" is replaced by "Symbol_Circle and Symbol_Circle_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.9.

In Table 3.3.22.2.1.9, Symbol_Circle and Symbol_Circle_Interactive, for Name "ItemType" added "A661_Symbol_Circle_Interactive" in Description and Value/Range.

3.3.22.2.1.10 Symbol_Rotated and Symbol_Rotated_Interactive

"Symbol_Rotated" is replaced by "Symbol_Rotated and Symbol_Rotated_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.10.

In Table 3.3.22.2.1.10, Symbol_Rotated and Symbol_Rotated_Interactive, for Name "ItemType" added "A661_Symbol_Rotated_Interactive" in Description and Value/Range.

3.3.22.2.1.11 Symbol_Runway and Symbol_Runway_Interactive

"Symbol_Runway" is replaced by "Symbol_Runway and Symbol_Runway_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.11.

In Table 3.3.22.2.1.11, Symbol_Runway and Symbol_Runway_Interactive, for Name "ItemType" added "A661_Symbol_Runway_Interactive" in Description and Value/Range.

3.3.22.2.1.12 Filled_Poly_Start and Filled_Poly_Start_Interactive

"Filled_Poly_Start" is replaced by "Filled_Poly_Start and Filled_Poly_Start_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.12.

In Table 3.3.22.2.1.12, Filled_Poly_Start and Filled_Poly_Start_Interactive:

- Name "ItemType" added "A661_Filled_Poly_Start_Interactive" in Description and Value/Range
- Name "FillStyleIndex" Description and Value/Range updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.13 Symbol_Oval and Symbol_Oval_Interactive

"Symbol_Oval" is replaced by "Symbol_Oval and Symbol_Oval_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.13.

In Table 3.3.22.2.1.13, Symbol_Oval and Symbol_Oval_Interactive:

- Name "ItemType" added "A661_Symbol_Oval_Interactive" in Description and Value/Range
- Name "FillStyleIndex" Description and Value/Range updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.14 Item_Synchronization

In paragraph 2, "MapHorzItemList" was changed to "MapHorz_ItemList" and "MapVertItemList" was changed to "MapVert_ItemList".

3.3.22.2.1.15 Symbol_Target and Symbol_Target_Interactive

"Symbol_Target" is replaced by "Symbol_Target and Symbol_Target_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.15.

In Table 3.3.22.2.1.15, Symbol_Target and Symbol_Target_Interactive, Name "ItemType" added "A661_Symbol_Target_Interactive" in Description and Value/Range.

3.3.22.2.1.16 Triangle_Strip_Start and Triangle_Strip_Start_Interactive

“Triangle_Strip_Start” was replaced by “Triangle_Strip_Start and Triangle_Strip_Start_Interactive” throughout the section, with the exception of inside Table 3.3.22.2.1.16.

In Table 3.3.22.2.1.16, Triangle_Strip_Start and Triangle_Strip_Start_Interactive, Name “ItemType” added “A661_ Triangle_Strip_Start_Interactive” in Description and Value/Range.

3.3.22.2.1.17 Triangle_Segment and Triangle_Segment_Interactive

“Triangle_Segment” was replaced by “Triangle_Segment and Triangle_Segment_Interactive” throughout the section, with the exception of inside Table 3.3.22.2.1.17.

In Table 3.3.22.2.1.17, Triangle_Segment and Triangle_Segment_Interactive:

- Name “ItemType” added “A661_Triangle_Segment_Interactive” in Description and Value/Range
- “FillStyleIndex” Description and Value/Range, was updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.18 Triangle_Segment_Double and Triangle_Segment_Double_Interactive

“Triangle_Segment_Double” was replaced by “Triangle_Segment_Double and Triangle_Segment_Double_Interactive” throughout the section, with the exception of inside Table 3.3.22.2.1.18.

In Table 3.3.22.2.1.17, Triangle_Segment_Double and Triangle_Segment_Double_Interactive:

- Name “ItemType” added “A661_Triangle_Segment_Double_Interactive” in Description and Value/Range
- “FillStyleIndex” Description and Value/Range, was updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.19 Triangle_End and Triangle_End_Interactive

“Triangle_End” was replaced by “Triangle_End and Triangle_End_Interactive” throughout the section, with the exception of inside Table 3.3.22.2.1.19.

In Table 3.3.22.2.1.19, Triangle_End and Triangle_End_Interactive:

- Name “ItemType” added “A661_Triangle_End_Interactive” in Description and Value/Range
- “FillStyleIndex” Description and Value/Range, was updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.20 Triangle_End_Double and Triangle_End_Double_Interactive

“Triangle_End_Double” was replaced by “Triangle_End_Double and Triangle_End_Double_Interactive” throughout the section, with the exception of inside Table 3.3.22.2.1.20.

In Table 3.3.22.2.1.20, Triangle_End_Double and Triangle_End_Double_Interactive:

- Name “ItemType” added “A661_Triangle_End_Double_Interactive” in Description and Value/Range
- “FillStyleIndex” Description and Value/Range, was updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.21 Triangle_Fan_Start and Triangle_Fan_Start_Interactive

"Triangle_Fan_Start" was replaced by "Triangle_Fan_Start and Triangle_Fan_Start_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.21.

In Table 3.3.22.2.1.20, Triangle_End_Double and Triangle_End_Double_Interactive, Name "ItemType" added "A661_Triangle_Fan_Start_Interactive" in Description and Value/Range.

3.3.22.2.1.22 Legend_Anchor_Rotated

"Legend Anchor Rotated" was corrected to "Legend_Anchor_Rotated" throughout the section.

3.3.22.2.1.24 Symbol_Rectangle and Symbol_Rectangle_Interactive

"Symbol_Rectangle" was replaced by "Symbol_Rectangle and Symbol_Rectangle_Interactive" throughout the section, with the exception of inside Table 3.3.22.2.1.24.

In Table 3.3.22.2.1.24, Symbol_Rectangle and Symbol_Rectangle_Interactive:

Name "ItemType" added "A661_Symbol_Rectangle_Interactive" in Description and Value/Range

"FillStyleIndex" Description and Value/Range, was updated to add a reference to Section 3.1.3.3.1

3.3.22.2.1.25 Legend_Combo

Table 3.3.22.2.1.25-1, was added.

Figure 3.3.22.2.1.25, Legend_Combo, was deleted.

In Table 3.3.22.2.1.25-2, Legend_Combo, for Name "LegendString," the Type, Size (bits), and Description and Value/Range were changed.

3.3.22.2.1.26 Vertex_Render_Start

"Vertex Render Start" was corrected to "Vertex_Render_Start" throughout the section.

3.3.22.2.1.27 Vertex_Render_Segment

"Vertex Render Segment" was corrected to "Vertex_Render_Segment" throughout the section.

In Table 3.3.22.2.1.27, Vertex_Render_Segment, for Name "Indices", Type was changed from "ushort" to "ushort x 8".

3.3.22.2.1.28 Vertex_Render_End

"Vertex Render End" was corrected to "Vertex_Render_End" throughout the section.

In Table 3.3.22.2.1.28, Vertex_Render_End:

For Name "NumIndices", Description and Value/Range was changed to read "Number of indices used in this list"

For Name "Indices", Type was changed from "ushort" to "ushort x 8" and Description and Value/Range was changed from "Up to 8 indices" to "Unused indeces set to 0"

3.3.22.2.1.29 Symbol_Parkable and Symbol_Parkable_Interactive

"Parkable Symbol and Parkable_Symbol_Interactive" was corrected to "Symbol_Parkable and Symbol_Parkable_Interactive" throughout the section.

In Table 3.3.22.2.1.29, Symbol_Parkable and Symbol_Parkable_Interactive, Name "ItemType" added "A661_Symbol_Parkable" in Description and Value/Range.

3.3.22.2.1.30 Symbol_Parkable_Rotated and Symbol_Parkable_Rotated_Interactive

This section was added to describe the Symbol_Parkable_Rotated and Symbol_Parkable_Rotated_Interactive.

"ParkableSymbol" was replaced by "Parkable_Symbol" throughout the section.

3.3.22.2.1.31 Symbol_Target_Parkable and Symbol_Target_Parkable_Interactive

This section was added to describe the Symbol_Target_Parkable and Symbol_Target_Parkable_Interactive.

"ParkableSymbol" was replaced by "Symbol" throughout the section.

3.3.22.2.1.32 Parking_Line_Start

Parking Line Start" was replaced by "Parking_Line_Start" throughout the section.

In Table 3.3.22.2.1.32, Parking_Line_Start, Parameter "Pad" is changed to "UnusedPad".

3.3.22.2.1.33 Parking_Line_End

In Table 3.3.22.2.1.33, Parking_Line_End, Parameter "Pad" is changed to "UnusedPad".

3.3.22.2.1.35 Legend_Readout_Format_String

This section was added to describe the Legend_Readout_Format_String.

3.3.22.2.1.36 Legend_Readout

This section was added to describe the Legend_Readout.

3.3.22.2.1.37 Legend_Readout_Combo

This section was added to describe the Legend_Readout_Combo.

3.3.22.3 MapHorz_ItemList Interactive Map Items

Interactive map items description and list was deleted.

3.3.24 MapHorz_Source

Added clarifying language in the Restriction section.

In Table 3.3.24-1, MapHorz_Source Parameters, Parameter "EventFlag" Description was clarified.

In Table 3.3.24-2b, MapDataFormat Values, in the columns "Alignment of +Y Axis" and "Positive Orientation," the X and Y axes were defined. For A661_MDF_LAT_LONG, the Origin was also defined.

Notes were added at the end for the table.

3.3.25 MapHorz

Table 3.3.25-2b, MapHorz Event Structures A661_EVT_ITEM_SYNCRONIZATION, MapHorz_ItemList was changed from MapHorzItemList.

3.3.26 MaskContainer

The Description was clarified.

In Table 3.3.26-1, MaskContainer Parameters, Parameter "MotionAllowed" was added.

In Table 3.3.26-2, MaskContainer Creation Structure, CreateParameterBuffers "MotionAllowed" and "Unused pad" were added.

In Table 3.3.26-3, MaskContainer Runtime Modifiable Parameters, Name of the Parameter to be Set "PosX," "PosY" and "PosX, PosY" were added.

3.3.27 Panel

Category "Dynamic Motion" was deleted.

3.3.28 PicturePushButton

In Table 3.3.28-1, PicturePushButton Parameters:

Parameters "PosX", "PosY", "SizeX", "SizeY", "Alignment", and "PicturePosition" Change values were changed from "D" to "DR"

Parameter "MaxStringLength" Description was clarified

In Table 3.3.28-4, Picture PushButton Runtime Modifiable Parameters, the following Parameters were added: "PosX", "PosY", "PosX, PosY", "SizeX", "SizeY", "SizeX, SizeY", "Alignment", and "PicturePosition".

3.3.29 PictureToggleButton

In Table 3.3.29-1, PicturePushButton Parameters:

Parameters "PosX", "PosY", "SizeX", "SizeY", "AlternateFlag", "Alignment", and "PicturePosition" Change values were changed from "D" to "DR"

Parameter "MaxStringLength" Description was clarified

In Table 3.3.29-4, PictureToggleButton Runtime Modifiable Parameters, the following Parameters were added: "PosX", "PosY", "PosX, PosY", "SizeX", "SizeY", "SizeX, SizeY", "Alignment", "PicturePosition", and "AlternateFlag".

3.3.30 PopUpPanel

In Table 3.3.30-1, PopUpPanel Parameters, Parameters "PosX", "PosY", "SizeX", "SizeY", "UAPositionFlag", and "AutomaticClosure" Change values were changed from "D" to "DR".

In Table 3.3.30-4, PopUpPanel Runtime Modifiable Parameters, the following Parameters were added: "PosX", "PosY", "PosX, PosY", "SizeX", "SizeY", "SizeX, SizeY", "UAPositionFlag", and "AutomaticClosure".

3.3.31 PopUpMenu

In the Description, the logic used is changed to "implementation dependent".

In Table 3.3.31-1, PopUpMenu Parameters:

Parameters "PosX", "PosY", "SizeX", "SizeY", and "OpeningMode" Change values were changed from "D" to "DR"

Parameter "MaxStringLength" Description was clarified

In Table 3.3.31-2, PopUpMenu Creation Structure, Parameter "MaxStringLength" Description was clarified.

In Table 3.3.31-4, PopUpMenu Runtime Modifiable Parameters, the following Parameters were added: "PosX", "PosY", "PosX, PosY", "SizeX", "SizeY", "SizeX, SizeY", and "OpeningMode".

3.3.32 PopUpMenuButton

In Table 3.3.32-1, PopUpMenuButton Parameters:

Parameters "PosX", "PosY", "SizeX", "SizeY", "Alignment", "PicturePosition", "PopUpPosX", "PopUpPosY", "PopUpSizeX", "PopUpSizeY", and "OpeningMode" Change values were changed from "D" to "DR"

Parameter “MaxStringLength” and “MaxStringLengthPopUp” Descriptions were clarified

In Table 3.3.32-4, PopUpMenuButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “OpeningMode”, “PicturePosition”, “PopUpPosX”, “PopUpPosY”, “PopUpSizeX”, and “PopUpSizeY”.

3.3.33 PushButton

In Table 3.3.33-1, PushButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, and “Alignment” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.3.33-4, PushButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, and “Alignment”.

3.3.36 ScrollPanel

In the Description, the style to be used is changed to “implementation dependent”.

In Table 3.3.36-1, ScrollPanel Parameters, Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “LineDeltaX”, “LineDeltaY”, “PageDeltaX”, “PageDeltaY”, “HomeX”, “HomeY”, SizeXSheet”, “SizeYSheet”, “HorizontalScroll”, and “VerticalScroll” Change values were changed from “D” to “DR”.

In Table 3.3.36-4, ScrollPanel Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “HomeX”, “HomeY”, “LineDeltaX”, “LineDeltaY”, “PageDeltaX”, “PageDeltaY”, “HorizontalScroll”, and “VerticalScroll”.

3.3.37 ScrollList

In Table 3.3.37-1, ScrollList Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “Alignment”, and “VerticalScroll” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.3.37-5, ScrollList Runtime Modifiable Parameters:

The following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, and “VerticalScroll”

Parameter “LabelStringArray [up to MaxNumberOfEntries] and EnableArray [up to MaxNumberOfEntries]” Type was changed to {string}+

3.3.38 Symbol

Category “Dynamic Motion” was deleted.

In Table 3.3.38-1, Symbol Parameters, Parameter “ColorIndex” Description was expanded to add a reference to Section 3.1.3.3.1.

3.3.39 TabbedPanel

In Table 3.3.39-1, TabbedPanel Parameters:

Parameters “Alignment”, “InsetSize”, and “PicturePosition” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.3.39-3, TabbedPanel Runtime Modifiable Parameters, the following Parameters were added: “Alignment”, “InsetSize”, and “PicturePosition”.

3.3.40 TabbedPanelGroup

A Note was added to the Description concerning the ActiveTabbedPanelID.

In Table 3.3.40-1, TabbedPanelGroup Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, and “TabPosition” Change values were changed from “D” to “DR”

For Parameter “AutomaticInsetSizeFlag”, in the Description, “ButtonSize” was changed to “InsetSize.”

In Table 3.3.40-4, TabbedPanelGroup Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, and “TabPosition”.

3.3.41 ToggleButton

In Table 3.3.41-1, ToggleButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “AlternateFlag”, and “Alignment” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.3.41-2, ToggleButton Creation Structure, Parameter “LabelString” Description was clarified.

In Table 3.3.41-4, ToggleButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “AlternateFlag”, and “Alignment”.

3.4.1.1 MapGrid A661_ParameterStructure Specifics

In Table 3.4.1.1-1, A661_ParameterStructure_BufferOfFillStyles, Field “ParameterValue” Description was expanded to add a reference to Section 3.1.3.3.1.

3.4.1.2 Fill Style Index Values

The definition paragraph was deleted and replaced with a reference to Section 3.1.3.3.1.

3.4.2 ExternalSource

Category “Dynamic Motion” was deleted.

In the Description, the Note is corrected for clarity.

In Table 3.4.2-1, ExternalSource Parameters:

Parameter “MotionAllowed” was added

“Specific Parameters” are limited to “SourceReference”, “SourceX”, “SourceY”, “SourceDX”, “SourceDY”, and “StyleSet”

In Table 3.4.2-2, ExternalSource Creation Structure, CreateParameterBuffer “MotionAllowed” was added.

In Table 3.4.2-3, ExternalSource Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, SizeX”, “SizeY”, “SizeX, SizeY”, and “SourceReference”.

3.4.3 MapVert

In Table 3.4.3-2b – MapVert Event Structures: A661_EVT_ITEM_SYNCHRONIZATION, for EventStructure “LinkedIdent”, in the Value/Description, “MapVertItemList” was replaced with “MapVert_ItemList”.

3.4.4 MapVert_Source

The Restrictions description was clarified.

3.4.5 MapVert_ItemList

In Table 3.4.5-1, MapVert_ItemList Parameters, Parameter “MaxItemSize” was added.

In Table 3.4.5-2a, MapVert_ItemList Creation Structure, CreateParameterBuffers “MaxItemSize” and “UnusedPad” were added.

3.4.5.1 MapVert_ItemList Standard Items Description

For Table 3.4.5.1, MapVert_ItemList Standard Items Description:

The column “Size in Words” was added

Name of Item “LEGEND_READOUT_HIGHLIGHT” was deleted

The following Name of Items were added:

“FILLED_POLY_START_INTERACTIVE”, “LEGEND”, “LEGEND_COMBO”, “LEGEND_READOUT”, “LEGEND_READOUT_FORMAT_STRING”, “LEGEND_READOUT_COMBO”, “LINE_START_INTERACTIVE”, “LINE_SEGMENT_INTERACTIVE”, “SYMBOL_GENERIC_INTERACTIVE”, “SYMBOL_ROTATED_INTERACTIVE”, “SYMBOL_RUNWAY_INTERACTIVE”, “SYMBOL_TARGET_INTERACTIVE”, “SYMBOL_RECTANGLE_INTERACTIVE”, “TRIANGLE_STRIP_START_INTERACTIVE”, “TRIANGLE_FAN_START_INTERACTIVE”, “TRIANGLE_SEGMENT_INTERACTIVE”, “TRIANGLE_SEGMENT_DOUBLE_INTERACTIVE”, “TRIANGLE_END_INTERACTIVE”, “TRIANGLE_END_DOUBLE_INTERACTIVE”

3.4.5.2.1 Item Structures

The Commentary was changed to clarify use of a maximum size of a MIL item or by using the MaxItemSize parameter.

3.4.5.2.1.4 Line_Start and Line_Start_Interactive

“Line_Start” was replaced by “Line_Start and Line_Start_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.4.

In Table 3.4.5.2.1.4, Line_Start and Line_Start_Interactive, Name “ItemType” added “A661_Line_Start_Interactive” in Description and Value/Range.

3.4.5.2.1.5 Line_Segment and Line_Segment_Interactive

“Line_Segment” was replaced by “Line_Segment and Line_Segment_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.5.

In Table 3.4.5.2.1.5, Line_Segment and Line_Segment_Interactive, Name “ItemType” added “A661_Line_Segment_Interactive” in Description and Value/Range.

3.4.5.2.1.7 Symbol_Generic and Symbol_Generic_Interactive

“Symbol_Generic” was replaced by “Symbol_Generic and Symbol_Generic_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.7.

In Table 3.4.5.2.1.7, Symbol_Generic and Symbol_Generic_Interactive:

Name “ItemType” added “A661_Symbol_Generic_Interactive” in Description and Value/Range

Names “RelativePosition” and “UnusedPad” were changed to Type uchar

Name “SymbolType” was changed to Type ushort

3.4.5.2.1.8 Symbol_Runway and Symbol_Runway_Interactive

“Symbol_Runway” was replaced by “Symbol_Runway and Symbol_Runway_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.8.

In Table 3.4.5.2.1.8, Symbol_Runway and Symbol_Runway_Interactive, Name “ItemType” added “A661_Symbol_Runway_Interactive” in Description and Value/Range.

3.4.5.2.1.9 Filled_Poly_Start and Filled_Poly_Start_Interactive

“Filled_Poly_Start” was replaced by “Filled_Poly_Start and Filled_Poly_Start_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.9.

In the definition paragraph, “airframe manufacturer/system integrator” is replaced with “OEM”.

In Table 3.4.5.2.1.9, Filled_Poly_Start and Filled_Poly_Start_Interactive:

Name “ItemType” added “A661_Filled_Poly_Start_Interactive” in Description and Value/Range

For Name “FillStyleIndex” in Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.11 Symbol_Rotated and Symbol_Rotated_Interactive

“Symbol_Rotated” was replaced by “Symbol_Rotated and Symbol_Rotated_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.11.

In Table 3.4.5.2.1.11, Symbol_Rotated and Symbol_Rotated_Interactive, Name “ItemType” added “A661_Symbol_Rotated_Interactive” in Description and Value/Range.

3.4.5.2.1.12 Triangle_Strip_Start and Triangle_Strip_Start_Interactive

Variable names were corrected in the two introductory paragraphs.

“Triangle_Strip_Start” was replaced by “Triangle_Strip_Start and Triangle_Strip_Start_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.12.

In Table 3.4.5.2.1.1, Triangle_Strip_Start and Triangle_Strip_Start_Interactive, Name “ItemType” added “A661_Triangle_Strip_Start_Interactive” in Description and Value/Range.

3.4.5.2.1.13 Triangle_Segment and Triangle_Segment_Interactive

“Triangle Segment” was replaced by “Triangle_Segment and Triangle_Segment_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.13.

In Table 3.4.5.2.1.13, Triangle_Segment and Triangle_Segment_Interactive:

Name “ItemType” added “A661_Triangle_Segment_Interactive” in Description and Value/Range

Name “FillStyleIndex” Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.14 Triangle_Segment_Double and Triangle_Segment_Double_Interactive

"Triangle Segment Double" was replaced by "Triangle_Segment_Double and Triangle_Segment_Double_Interactive" throughout the section, with the exception of inside Table 3.4.5.2.1.14.

In Table 3.4.5.2.1.14, Triangle_Segment_Double and Triangle_Segment_Double_Interactive:

Name "ItemType" added "A661_Triangle_Segment_Double_Interactive" in Description and Value/Range

Name "FillStyleIndex" Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.15 Triangle_End and Triangle_End_Interactive

"Triangle End" was replaced by "Triangle_End and Triangle_End_Interactive" throughout the section, with the exception of inside Table 3.4.5.2.1.15.

In Table 3.4.5.2.1.15 Triangle_End and Triangle_End_Interactive:

Name "ItemType" added "A661_Triangle_End_Interactive" in Description and Value/Range

Name "FillStyleIndex" Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.16 Triangle_End_Double and Triangle_End_Double_Interactive

"Triangle End Double" was replaced by "Triangle_End_Double and Triangle_End_Double_Interactive" throughout the section, with the exception of inside Table 3.4.5.2.1.16.

In Table 3.4.5.2.1.16 Triangle_End_Double and Triangle_End_Double_Interactive:

Name "ItemType" added "A661_Triangle_End_Double_Interactive" in Description and Value/Range

Name "FillStyleIndex" Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.17 Triangle_Fan_Start and Triangle_Fan_Start_Interactive

Variable names were corrected throughout the section.

"Triangle Fan Start" was replaced by "Triangle_Fan_Start and Triangle_Fan_Start_Interactive" throughout the section, with the exception of inside Table 3.4.5.2.1.17.

In Table 3.4.5.2.1.17, Triangle_Fan_Start and Triangle_Fan_Start_Interactive, Name "ItemType" added "A661_Triangle_Fan_Start_Interactive" in Description and Value/Range.

3.4.5.2.1.18 Legend_Anchor_Rotated

"Legend Anchor Rotated" was changed to "Legend_Anchor_Rotated" throughout the section.

3.4.5.2.1.20 Symbol_Target and Symbol_Target_Interactive

"Symbol Target" was replaced by "Symbol_Target and Symbol_Target_Interactive" throughout the section, with the exception of inside Table 3.4.5.2.1.20.

In Table 3.4.5.2.1.20, Symbol_Target and Symbol_Target_Interactive, Name "ItemType" added "A661_Symbol_Target_Interactive" in Description and Value/Range.

3.4.5.2.1.21 Symbol_Rectangle and Symbol_Rectangle_Interactive

“Symbol_Rectangle” was replaced by “Symbol_Rectangle and Symbol_Rectangle_Interactive” throughout the section, with the exception of inside Table 3.4.5.2.1.21.

In Table 3.4.5.2.1.21 Symbol_Rectangle and Symbol_Rectangle_Interactive:

Name “ItemType” added “A661_Symbol_Rectangle_Interactive” in Description and Value/Range

Name “FillStyleIndex” Description and Value/Range, was expanded to add a reference to Section 3.1.3.3.1

3.4.5.2.1.22 Legend_Combo

Table 3.4.5.2.1.22-1, Legend_Combo, was added.

Figure 3.4.5.2.1.22, Legend_Combo was deleted.

In Table 3.4.5.2.1.22-2, Legend_Combo, for Name “Legend_String”, Type, Size (bits), and Description and Value/Range were clarified.

3.4.5.2.1.23 Legend_Readout_Format_String

This section was added to describe the Legend_Readout_Format_String.

3.4.5.2.1.24 Legend_Readout

This section was added to describe the Legend_Readout.

3.4.5.2.1.25 Legend_Readout_Combo

This section was added to describe the Legend_Readout_Combo.

3.4.5.3 MapVert_ItemList Interactive Map Items

Interactive map items description and list was deleted.

3.4.6 EditBoxMultiLine

In Table 3.4.6-1, EditBoxMultiLine Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “Alignment”, and “VerticalScroll” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

Parameter “StartCursorPos” Description was expanded to add a reference to Section 3.1.3.3

In Table 3.4.6-7, EditBoxMultiLine Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, and “VerticalScroll”.

3.4.7 ComboBoxEdit

In Table 3.4.7-1, ComboBoxEdit Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “SelectingAreaWidth”, “SelectingAreaHeight”, “OpeningMode”, and “Alignment” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

Parameter “StartCursorPos” Description was expanded to add a reference to Section 3.1.3.3

In Table 3.4.7-8, ComboBoxEdit Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “OpeningMode”, “SelectingAreaWidth”, and “SelectingAreaHeight”.

3.4.8 MenuBar

In Table 3.4.8-1, MenuBar Parameters, Parameters “PosX”, “PosY”, “Horizontal”, “ButtonPos”, and “ButtonSize” Change values were changed from “D” to “DR”.

In Table 3.4.8-3, MenuBar Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “Horizontal”, “ButtonPos”, and “ButtonSize”.

3.5.1 MutuallyExclusiveContainer

Category “Dynamic Motion” was deleted.

3.5.4 Slider

In Table 3.5.4-1, Slider Parameters, Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “MinValue”, “MaxValue”, “MajorTickInterval”, “ShowMajorLabels”, “Alignment”, “Orientation”, “LsbMultiple”, “MinorTickMultiple”, and “MajorTickReference” Change values were changed from “D” to “DR”.

For Table 3.5.4-4, Slider Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “MinValue”, “MaxValue”, “MajorTickInterval”, “MinorTickMultiple”, “MajorTickReference”, “ShowMajorLabels”, “Orientation”, “Alignment”, and “LsbMultiple”.

3.5.5 PictureAnimated

In Table 3.5.5-1, PictureAnimated Parameters, Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “IndexOfFrequency”, and “LoopFlag” Change values were changed from “D” to “DR”.

In Table 3.5.5-3, PictureAnimated Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “IndexOfFrequency”, and “LoopFlag”.

3.5.6 SymbolAnimated

In Table 3.5.6-1, SymbolAnimated Parameters:

Parameter “ColorIndex” was expanded to add a reference to Section 3.1.3.3

Parameters “IndexOfFrequency” and “LoopType” Change values were changed from “D” to “DR”

In Table 3.5.6-3, SymbolAnimated Runtime Modifiable Parameters, the following Parameters were added: “IndexOfFrequency” and “LoopFlag”.

3.5.7 SelectionListButton

In Table 3.5.7-1, SelectionListButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “SelectingAreaWidth”, “SelectingAreaHeight”, “OpeningMode”, and “Alignment” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.5.7-4, SelectionListButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “SelectingAreaWidth”, “SelectingAreaHeight”, “OpeningMode”, and “Alignment”.

3.6.1 EditBoxNumericBCD

In Table 3.6.1-4, EditBoxNumericBCD Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “PositiveString”, “NegativeString”, “Alignment”, “LegendAreaSizeX”, “LegendPosition”, and “LegendRemoved” Change values were changed from “D” to “DR”

For Parameter “FormatString”, Examples were placed in the Description

For Parameters “MaxFormatStringLength”, “PositiveStringLength”, and “NegativeStringLength”, the maximum size in bytes was clarified

Table 3.6.1-5, Creation Structure, was renamed EditBoxNumericBCD Creation Structure.

In Table 3.6.1-10, EditBoxNumericBCD Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “PositiveString”, “NegativeString”, “LegendAreaSizeX”, “LegendPosition”, and “LegendRemoved”.

3.6.2 CursorRef

Category “Dynamic Motion” was deleted.

3.6.3 CursorOver

Category “Dynamic Motion” was deleted.

3.6.4 ShuffleToFitContainer

This section was deleted.

3.6.4 Focus Navigation Widgets

Renumbered from the former 3.6.5.

3.6.4.1 FocusLink

Renumbered from the former 3.6.5.1.

3.6.4.2 FocusIn

Renumbered from the former 3.6.5.2.

3.6.4.3 FocusOut

Renumbered from the former 3.6.5.3.

3.6.5 SizeToFitContainer

Renumbered from the former 3.6.6.

Category “Dynamic Motion” was deleted.

In Table 3.6.5-1, SizeToFitContainer Parameters, Parameter “ItemSpacing” Change value was changed from “D” to “DR”.

3.6.6 ShuffleToFitContainer

Renumbered from the former 3.6.7.

In Table 3.6.6-1, ShuffleToFitContainer Parameters, Parameter “ItemSpacing” Change value was changed from “D” to “DR”.

In Table 3.6.6-3, ShuffleToFitContainer Runtime Modifiable Parameters, the following Parameter was added: “ItemSpacing”.

3.7.1 SymbolPushButton

In Table 3.7.1-1, SymbolPushButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “Alignment”, and “SymbolPosition” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.7.1-4, SymbolPushButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, and “SymbolPosition”.

3.7.2 SymbolToggleButton

In Table 3.7.2-1, SymbolToggleButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “AlternateFlag”, “Alignment”, and “SymbolPosition” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.7.2-4, SymbolToggleButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “SymbolPosition” and “AlternateFlag”.

3.7.3 PopUpPanelButton

In Table 3.7.3-1, PopUpPanelButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, “MaxStringLength”, “Alignment”, “PicturePosition”, “PopUpPosX”, “PopUpPosY”, “PopUpSizeX”, “PopUpSizeY”, “UAPositionFlag”, and “AutomaticClosure” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.7.3-4, PopUpPanelButton Runtime Modifiable Parameters, the following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, “PicturePosition”, “PopUpPosX”, “PopUpPosY”, “PopUpSizeX”, “PopUpSizeY”, “UAPositionFlag”, and “AutomaticClosure”.

3.8.1 GpPolyLine

Category “Dynamic Motion” was deleted.

In Table 3.8.1-1, GpPolyline Parameters:

Parameter “Halo” and “Closed” Change values were changed from “D” to “DR”

Parameter “ColorIndex” Description was expanded to add a reference to Section 3.1.3.3.1

In Table 3.8.1-3, GpPolyline Runtime Modifiable Parameters, the following Parameters were added: “Halo” and “Closed”.

3.8.2 PagingContainer

Category “Dynamic Motion” was deleted.

In Table 3.8.2-1, PagingContainer Parameters, Parameters “WrappingType”, “PagingControlPosition”, “FinePageDelta”, and “CoarsePageDelta” Change values were changed from “D” to “DR”.

In Table 3.8.2-4, PagingContainer Runtime Modifiable Parameters, the following Parameters were added: “WrappingType”, “PagingControlPosition”, “FinePageDelta”, and “CoarsePageDelta”.

3.8.3 NumericReadout

Category “Dynamic Motion” was deleted.

In Table 3.8.3-1, NumericReadout Parameters:

Parameters “SizeX”, “SizeY”, “Alignment”, and “LegendAreaSizeX” Change values were changed from “D” to “DR”

Parameter “FormatString” changed Description to include “See section 3.2.6 “Formatted Numeric Values” for the formatting”

Parameter MaxFormatStringLength Description was clarified

In Table 3.8.3-3, NumericReadout Runtime Modifiable Parameters, the following Parameters were added: “SizeX”, “SizeY”, “SizeX, SizeY”, “Alignment”, and “LegendAreaSizeX”.

3.8.4 MapHorz_Container

Category “Dynamic Motion” was deleted.

In Table 3.8.4-5, MapHorz_Container Runtime Modifiable Parameters, for Parameter “X, Y”, Size (bits) was changed to “32x2.”

3.8.5 MapHorz_Panel

Category “Dynamic Motion” was deleted.

3.9.1 MultiStateButton

In Table 3.9.1-1, MultiStateButton Parameters:

Parameters “PosX”, “PosY”, “SizeX”, “SizeY”, and “Alignment” Change values were changed from “D” to “DR”

Parameter “MaxStringLength” Description was clarified

In Table 3.9.1-2, MultiStateButton Creation Structure, for Parameter LabelStringArray[NumberOfStates], Type was changed to “{string}+”.

In Table 3.9.1-4, MultiStateButton Runtime Modifiable Parameters:

The following Parameters were added: “PosX”, “PosY”, “PosX, PosY”, “SizeX”, “SizeY”, “SizeX, SizeY”, and “Alignment”

Parameter “LabelStringArray” Type was changed to “{string}+”.

3.9.2 KeyboardArea

In Table 3.9.2-1, KeyboardArea Parameters, Parameters “MaxNumberOfKeys” and “KeyArray” Descriptions were clarified.

In Table 3.9.2-2, KeyboardArea Creation Structure, CreateParameterBuffer “KeyArray” Value/Range when necessary was clarified.

In Table 3.9.2-3, KeyboardArea Event Structures: A661_EVT_KEY, Event Structure “A661 Key code of the pressed key” Value/Description was clarified.

Keycode organization was added.

Table 3.9.2-4, KeyboardArea Standard Keycodes, was added.

3.9.4 MapHorz_VertexBuffer

In Table 3.9.4-2, MapHorz_VertexBuffer Creation Structure, for Parameter UnusedPad, Size (bits) was changed from “24” to “8”.

3.9.5 TouchArea

The description for Figure 3.9.5-1, TouchArea Event Generation Example, was clarified.

In Table 3.9.5-1, TouchArea Parameters, Parameter “MaxTouchPoints” Description was clarified.

In Table 3.9.5-2, TouchArea Creation Structure, for Parameter “Enable”, Value/Description added “A661_TRUE_WITH_VALIDATION.”

In Table 3.9.5-4, TouchArea Event: A661_EVT_TOUCH, for EventStructure “Timestamp[TouchPointsReported]”, Value/Description was expanded and clarified.

3.9.6 GestureArea

Table 3.9.6-1, GestureArea Report States, was added.

Table 3.9.6-2, GestureArea Report Modes, was added.

In Table 3.9.6-3, GestureArea Parameters, for Parameters “ReportMode” and “EventIdArray”, was clarified and renumbered.

In Table 3.9.6-4, GestureArea Creation Structure was renumbered:

- For Parameter “Enable”, “A661_TRUE_WITH_VALIDATION” was added to Value/Description
- For Parameter “ReportMode”, “A661_REPORT_ON_TRANSITION_WITH_CANDIDATE” and “A661_REPORT_ALL_WITH_CANDIDATE” was added to Value/Description
- For Parameter “EventIdArray[NumberOfEvents]”: Type was changed from “ushort” to “uchar”, Size (bits) was changed to “8*NumberOfEvents + Pad”, and Value/Description was clarified to read, “There are NumberOfEvents in the array. The complete array is followed by zero, one, two, or three bytes(s) of padding to be 32 bits aligned”

3.9.6.1.3 A661_GESTURE_PRESS_AND_HOLD

Added description of gesture candidate reporting.

Added Figure 3.9.6-7, Press and Hold Reporting Mode State Transition.

In Table 3.9.6.1.3-1, Gesture Event: A661_EVT_GESTURE_PRESS_AND_HOLD, for Parameter “State”, “A661_GESTURE_CANDIDATE” was added to Value/Description.

3.9.6.1.4 A661_GESTURE_PINCH

Added paragraphs on gesture candidate reporting and gesture recognition.

Added title for Figure 3.9.6-9.

In Table 3.9.6.1.4-1, Gesture Event: A661_EVT_GESTURE_PINCH, for Event Structure “State”, “A661_GESTURE_CANDIDATE” was added to Value/Description.

3.9.6.1.5 A661_GESTURE_ROTATE

Added paragraphs on gesture candidate reporting and gesture recognition.

Added title for Figure 3.9.6-12.

In Table 3.9.6.1.5-1, Gesture Event: A661_EVT_GESTURE_ROTATE, for Event Structure “State”, “A661_GESTURE_CANDIDATE” was added to Value/Description.

3.9.6.1.6 A661_GESTURE_DRAG

Added paragraphs on gesture candidate reporting and gesture recognition.

Added title for Figure 3.9.6-15.

In Table 3.9.6.1.6-1, Gesture Event: A661_EVT_GESTURE_DRAG, for Event Structure “State”, “A661_GESTURE_CANDIDATE” was added to Value/Description.

3.9.6.1.9 A661_GESTURE_HOLD

This section was added to describe the A661_GESTURE_HOLD.

3.9.7 InkArea

In Table 3.9.7-1, InkArea Parameters:

- Parameters “ColorIndex” and “FillIndex” Descriptions were expanded to add a reference to Section 3.1.3.3.1
- Parameter “Filled” Change value was changed from “D” to “DR”
- Parameter “NextFocusedWidget” sentence referring to “MapHorz_Container” in Description was deleted

In Table 3.9.7-2, InkArea Creation Structure, CreateParameterBuffer “Enable” Value/Description added “A661_TRUE_WITH_VALIDATION.”

In Table 3.9.7-3, InkArea Runtime Modifiable Parameters, Parameter “Filled” was added.

3.9.8 AnimationOnParam

Notes were added concerning “ToValue” and “FromValue”, the “LookCapacities” attribute, and “A661_DEACTIVATE”.

Note concerning “LookCapacities” which was similar to a Previous Note concerning “LookCapacities” was deleted.

In Table 3.9.8-1, AnimationOnParam Parameters, for Parameter “AnimationState”, the following were added to the Description: “A661_DEACTIVATE”, “A661_FROM”, and “A661_TO”.

3.9.9 AnimationRotation

Notes were added for rotation values and “A661_DEACTIVATE”.

In Table 3.9.9-1, AnimationRotation Parameters.

For Parameters “CenterX” and “CenterY”, Descriptions were added: “(relative to the target widget coordinate system) ”

For Parameter “AnimationState”, the following were added to the Description: “A661_DEACTIVATE”, “A661_FROM”, and “A661_TO”

In Table 3.9.9-2, AnimationRotation Creation Structure, Parameter “UnusedPad” was added.

In Table 3.9.9-4, AnimationRotation Runtime Modifiable Parameters, Name of the Parameter to Set “FromAngle” and “ToAngle” Type was changed to “fr(180)”.

3.9.10 AnimationScale

Notes concerning scale values and “A661_DEACTIVATE” were added.

In Table 3.9.10-1, AnimationScale Parameters, for Parameter “AnimationState”, the following were added to the Description: “A661_DEACTIVATE”, “A661_FROM”, and “A661_TO”.

3.9.11 AnimationTranslation

Notes were added for translation values and “A661_DEACTIVATE”.

In Table 3.9.11-1, AnimationTranslation Parameters, for Parameter “AnimationState”, the following were added to the Description: “A661_DEACTIVATE”, “A661_FROM”, and “A661_TO”.

3.9.12 AnimationGroup

The state transition algorithm for the widget descriptions was expanded and clarified.

In Table 3.9.12-1, AnimationGroup Parameters:

For Parameter “Type”, “A661_SEQUENTIAL_FROM” was added to the Description

For Parameter “AnimationState”, the following were added to the Description: “A661_DEACTIVATE”, “A661_FROM”, and “A661_TO”

In Table 3.9.12-4, AnimationGroup Runtime Modifiable Parameters, Parameter “Duration” was deleted.

3.9.13 MapVert_Container

Category “Dynamic Motion” was deleted.

In Table 3.9.13-5, MapVert_Container Runtime Modifiable Parameters, Name of the Parameter to be Set “X, Y” Size (bits) was changed to “32x2.”

3.9.14 MapVert_Panel

Category “Dynamic Motion” was deleted.

3.9.15 MapBoundary

Category “Dynamic Motion” was deleted.

Description was changed to add an example using Parkable_Symbol and include new map items impacted by the MapBoundary widget:

PARKABLE_SYMBOL_ROTATED,
PARKABLE_SYMBOL_ROTATED_INTERACTIVE,
PARKABLE_SYMBOL_TARGET and
PARKABLE_SYMBOL_TARGET_INTERACTIVE.

In Table 3.9.15-1, MapBoundary Parameters, Parameters “BeginAngleArray [NumberOfEntries]” and “DistanceArray [NumberOfEntries]” descriptions were clarified.

3.9.16 DataConnector

In Table 3.9.16-2, DataConnector Creation Structure, CreateParameterBuffer, “DataLength” Type was changed to “ushort”.

3.9.18 FramingRefreshContainer

In Table 3.9.18-2, FramingRefreshContainer Creation Structure, CreateParameterBuffer “Enable” added “A661_TRUE_WITH_VALIDATION” in Value/Description.

3.10 Widgets Added for Supplement 7

This section, with added subsections, describes 3 new widgets added for Supplement 7: “ScaleContainer”, “Selector”, “Tree”, “MapExternalSource”, “GpTriangleFan”, and “GpTriangleStrip”.

3.10.1 ScaleContainer

This section added to describe the “ScaleContainer” widget.

3.10.2 Selector

This section added to describe the “Selector” widget.

3.10.3 Tree

This section added to describe the “Tree” widget.

3.10.4 MapExternalSource

This section added to describe the “MapExternalSource” widget.

3.10.5 GpTriangleFan

This section added to describe the “GpTriangleFan” widget.

3.10.6 GpTriangleStrip

This section added to describe the “GpTriangleStrip” widget.

4.5.4.4 Notification Structure

In Table 4.5.4.4-1, Layer_Notification_Structure, for A661_Layer_Notification_Structure, “A661_Connector_Data_Structure” was added.

4.6 ARINC 661 Keyword Values

In Table 4.6-7, Widgets (16 bits), the following widgets were added: “A661_GP_TRIANGLE_FAN”, “A661_GP_TRIANGLE_STRIP”, “A661_MAP_EXTERNAL_SOURCE”, “A661_SCALE_CONTAINER”, and “A661_TREE”.

In Table 4.6-8, Parameter Types:

“A661_BOUNDARY_ARRAY” was deleted

“A661_ENABLE_HANDLER” was changed

The following Parameters were added: “A661_ALIGNMENT“, “A661_ALTERNATE_FLAG”, “A661_AUTO_CLOSURE”, “A661_BUTTON_POS”, “A661_BUTTON_SIZE”, “A661_CLOSED”, “A661_COARSE_PAGE_DELTA”, “A661_DIRECTION”, “A661_FILLED”, “A661_FINE_PAGE_DELTA”, “A661_HALO”, “A661_HOME_X”, “A661_HOME_Y”, “A661_HORIZONTAL_SCROLL”, “A661_INDEX_OF_FREQUENCY”, “A661_INSET_SIZE“, “A661_ITEM_SPACING“, “A661_LAYOUT”, “A661_LEGEND_ALIGNMENT”, “A661_LEGEND_AREA_SIZE_X”, “A661_LEGEND_REMOVED”, “A661_LINE_DELTA_X”, “A661_LINE_DELTA_Y”, “A661_LSB_MULTIPLE”, “A661_MAJOR_TICK_INTERVAL”, “A661_MAJOR_TICK_REFERENCE”, “A661_MAX_VALUE”, “A661_MENU_HORIZONTAL”, “A661_MIN_VALUE”, “A661_MINOR_TICK_MULTIPLE”, “A661_NEGATIVE_STRING”, “A661_NUMBER_OF_PICTURES”, “A661_NUMBER_OF_SYMBOLS”, “A661_OPENING_MODE”, “A661_OPEN_ENTRY_ARRAY”, “A661_PAGE_CONTROL_POSITION”, “A661_PAGE_DELTA_X”, “A661_PAGE_DELTA_Y”, “A661_PAGE_WRAP_TYPE”, “A661_PARENT_ENTRY_ARRAY”, “A661_PICTURE_POSITION”, “A661_POPUP_POS_X”, “A661_POPUP_POS_Y”, “A661_POPUP_SIZE_X”, “A661_POPUP_SIZE_Y”, “A661_POSITIVE_STRING”, “A661_REORDER”, “A661_SCALE_X”, “A661_SCALE_Y”, “A661_SELECTING_HEIGHT“, “A661_SELECTING_WIDTH“, “A661_SHEET_SIZE_X“, “A661_SHEET_SIZE_Y”, “A661_SHOW_MAJOR_TICKS”, “A661_SOURCE_REF”, “A661_SYMBOL_POSITION“, “A661_TAB_POSITION“, “A661_TREE_NODE_STYLE_ARRAY”, “A661_UA_POSITION_FLAG”, “A661_VERTICAL_SCROLL”, “A661_WORD_WRAP”, and “A661_WRAP_STYLE”

In Table 4.6-9, Event Types:

The following Parameters were deleted: “A661_EVT_CLICKED” “A661_EVT_DOUBLE_CLICKED”, “A661_EVT_PRESSED”, “A661_EVT_RELEASED”, and “A661_EVT_RIGHT_CLICKED”

The following Parameters were added: “A661_EVT_CURSOR_ENTRY_EVENT”, “A661_EVT_GESTURE_HOLD”, “A661_EVT_GESTURE_PINCH”, “A661_EVT_GESTURE_PRESS_AND_HOLD”, “A661_EVT_GESTURE_ROTATE”, “A661_EVT_MAPHORZ_ORIENTATION”, “A661_EVT_MAPHORZ_PROJECTION_REFERENCE”, “A661_EVT_MAPHORZ_RANGE”, “A661_EVT_MAPVERT_PROJECTION_REFERENCE”, “A661_EVT_MAPVERT_RANGE_X”, “A661_EVT_MAPVERT_RANGE_Y”,

"A661_EVT_TREE_ENTRY_OPEN", and
"A661_EVT_TREE_ENTRY_CLOSE"

In Table 4.6-11, Integer Constant Values:

The following Parameter was deleted: "A661_REPORT_ON_UA"

The following Parameters were added: "A661_DEACTIVATE", "A661_END", "A661_FROM", "A661_GESTURE_CANDIDATE", "A661_GESTURE_HOLD", "A661_HYPHENATE", "A661_LEGEND_READOUT", "A661_LEGEND_READOUT_COMBO", "A661_LEGEND_READOUT_FORMAT_STRING", "A661_PRESERVE_CHARACTER", "A661_PRESERVE_WORD", "A661_REPORT_ALL_WITH_CANDIDATE", "A661_REPORT_ON_TRANSITION_WITH_CANDIDATE", " A661_REPORT_ON_CDS", "A661_SEQUENTIAL_FROM", "A661_SYMBOL_ROTATED_PARKABLE", "A661_SYMBOL_ROTATED_PARKABLE_INTERACTIVE", "A661_SYMBOL_TARGET_PARKABLE", "A661_SYMBOL_TARGET_PARKABLE_INTERACTIVE", "A661_TO", "A661_TO_BACK", "A661_TO_FRONT", "A661_RANGE", "A661_ORIENTATION", "A661_PROJECTION_REFERENCE", "A661_ARC", "A661_HORZ_LINE", and "A661_VERT_LINE"

In Table 4.6-12, Widget Extensions, he following Parameters were added: "A661_REORDER_EXTENSION", "A661_WORD_WRAP_EXTENSION", "A661_CURSOR_ENTRY_EVENTS_EXTENSION", "A661_MAPHORZ_COORD_SYSTEM_EVENT_EXTENSION", and "A661_MAPVERT_COORD_SYSTEM_EVENT_EXTENSION".

Table 4.6-15, Layouts identifiers (8 bits), was added.

5.1 Overview

Discussion of SYMBOL_TARGET relating to Length and Orientation was added.

5.2.1 Top Level Commands

Table 5.2.1, Symbol Library Summary, added.

5.2.2.1 Set Color

In Table 5.2.2.1, SET_COLOR Command Structure, Field Name "ColorIndex" Description was expanded to add a reference to Section 3.1.3.3.1.

5.2.2.3 Set Font

In Table 5.2.2.3, SET_FONT Command Structure, Field Name "ColorIndex" Description was expanded to add a refrerence to Section 3.1.3.3.1.

5.2.4.4 Crown

In Table 5.2.4.4, CROWN Command Structure, Field Name "Filled" Description added "A661_TRUE" and "A661_FALSE".

5.2.4.8 Rectangle

In Table 5.2.4.8, RECTANGLE Command Structure, Field Name "Filled" Description added "A661_TRUE" and "A661_FALSE".

5.2.4.9 Triangle

In Table 5.2.4.9, TRIANGLE Command Structure, Field Name "Filled" Description added "A661_TRUE" and "A661_FALSE".

5.2.4.10 Triangle_Fan

Sentence, “The TRIANGLE_FAN command does not correspond to any Gp widget.” was deleted.

5.2.4.11 Triangle_Strip

Sentence, “The TRIANGLE_STRIP command does not correspond to any Gp widget.” was deleted.

5.2.4.13 Size

This section deprecated by Supplement 7.

6.5 XML Definition Split with Layer and WidgetSet

This section added to describe the XML Definition Split with Layer and WidgetSet.

6.5.1 Defining and Referencing

This section added to describe and diagram XML data file splits at the Layer and WidgetSet levels.

6.5.1.1 Possible Children of Layer and WidgetSet Definitions

This section added to refer to compliance with other sections of this document.

6.5.1.2 Layer

This section added to define and describe Layer properties.

6.5.1.3 WidgetSet

This section added to define and describe WidgetSet properties.

6.5.1.3.1 WidgetSet Variables

This section added to describe WidgetSet Variables.

6.5.1.3.2 Widget References

This section added to describe referencing widgets inside a WidgetSet.

6.5.1.3.3 Focus Navigation Using Variables

This section added to define the logical focus navigation order for the children widgets.

6.5.2 Reference Expansion

This section added to describe the rules to be followed when Layer and WidgetSet references go through expansion.

6.5.2.1 Layer Reference

This section added to define Layer references.

6.5.2.2 WidgetSet Reference

This section added to describe WidgetSet definitions.

6.5.2.2.1 Widget Name

This section added to define the structure of the Widget Name.

6.5.2.2.2 WidgetIdent

This section added to define the WidgetIdent parameters of a widget.

6.5.2.2.3 WidgetSet Variables

This section added to define the value of WidgetSet variables.

6.5.2.2.4 Widget Positions

This section added to define the position of a widget.

6.5.2.2.5 Respecting Possible Children of Container Widget Rules

This section added to address Possible Children of Container Widgets.

6.5.2.2.6 Nesting of WidgetSets

This section added to describe nesting of WidgetSets.

8.3 Extensions Summary

This new section and Table 8.3 added to describe the Extensions Summary.

8.4 Possible Widgets for Extension

In Table 8.4, Possible Widgets for Extension:

The following Extensions were added: “ReorderExtension”, “WordWrapExtension”, “CursorEntryEventsExtension”, “MapHorzCoordSystemEventExtension”, and “MapVertCoordSystemEventExtension”

The following Widgets were added: “MapExternalSource”, “ScaleContainer”, “Selector”, and “Tree”

8.5.3 LegendStringAlignmentExtension

In Table 8.5.3-1, LegendStringAlignment Parameters, Parameter “LegendAlignment” Change value was changed from “D” to “DR”.

Table 8.5.3-3, LegendStringAlignment Runtime Modifiable Parameters, was added.

8.6.1 PictureExtension

A new case added for widgets which have an array of labels and a specific label.

In Table 8.6.1-1, PictureExtension Parameters, Parameter “PicturePosition” Change value was changed from “D” to “DR”.

In Table 8.6.1-3, PictureExtension Runtime Modifiable Parameters, Parameter “PicturePosition” was added.

8.6.2 SymbolExtension

A new case added for widgets which have an array of labels and a specific label.

In Table 8.6.2-1,SymbolExtension Parameters, Parameter “SymbolPosition” Change value was changed from “D” to “DR”.

In Table 8.6.2-3, SymbolExtension Runtime Modifiable Parameters, Parameter “SymbolPosition” was added.

8.6.6 MultiSelectionExtension

In Table 8.6.6-3, MultiSelectionExtension Runtime Modifiable Parameters, Parameters “SelectAll” and “UnselectAll” Size (bits) was set to “8”.

8.7 Widget Extensions Added for Supplement 7

This is a new section, added to address new widget extensions for Supplement 7.

8.7.1 ReorderExtension

This section added to describe the ReorderExtension.

8.7.2 WordWrapExtension

This section added to describe the WordWrapExtension.

8.7.3 CursorEntryEventsExtension

This section added to describe the CursorentryEventsExtension.

8.7.4 MapHorzCoordSystemEventExtension

This section added to describe the MapHorzCoordSystemEventExtension.

8.7.5 MapVertCoordSystemEventExtension

This section added to describe the MapVertCoordSystemEventExtension.

APPENDIX E – MAP MANAGEMENT TUTORIAL

E-2.1 Addressing MapItems for One Change

In Table E-2.1-1, A661_ParameterStructure for the SetParameter command for SYMBOL_GENERIC, for Parameter “ItemStructure”, “Latitude” was added to the Parameter value.

E-2.2 Addressing MapItems for Multiple Changes

In Table E-2.2-1, A661_ParameterStructure for the SetParameter command, for Parameter “ItemStructure”, “Latitude” was added to the Parameter value.

APPENDIX G – NEW WIDGET GUIDLINES

G-5 Widget Category

Category “Dynamic Motion” was deleted.

G-9 Runtime Modifiable Parameters

Runtime modifiable parameter point number three clarified.

G-11.1 Cases for Definition Time Only Parameters

The definition time only list was clarified.

ARINC Standard – Errata Report

1. Document Title

(Insert the number, supplement level, date of publication, and title of the document with the error)

2. Reference

Page Number: _____ Section Number: _____ Date of Submission: _____

3. Error

(Reproduce the material in error, as it appears in the standard.)

4. Recommended Correction

(Reproduce the correction as it would appear in the corrected version of the material.)

5. Reason for Correction *(Optional)*

(State why the correction is necessary.)

6. Submitter (Optional)

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

Please return comments to standards@sae-itc.org

Note: Items 2-5 may be repeated for additional errata. All recommendations will be evaluated by the staff. Any substantive changes will require submission to the relevant subcommittee for incorporation into a subsequent Supplement.

[To be completed by IA Staff]

Errata Report Identifier: _____ **Engineer Assigned:** _____

Review Status: _____

ARINC Project Initiation/Modification (APIM)

1.0 Name of Proposed Project APIM #: _____

(Insert name of proposed project.)

1.1 Name of Originator and/or Organization

(Insert name of individual and/or the organization that initiated the APIM)

2.0 Subcommittee Assignment and Project Support

2.1 Suggested AEEC Group and Chairman

(Identify an existing or new AEEC group.)

2.2 Support for the activity (as verified)

Airlines: (Identify each company by name.)
Airframe Manufacturers:
Suppliers:
Others:

2.3 Commitment for Drafting and Meeting Participation (as verified)

Airlines:
Airframe Manufacturers:
Suppliers:
Others:

2.4 Recommended Coordination with other groups

(List other AEEC subcommittees or other groups.)

3.0 Project Scope (why and when standard is needed)

3.1 Description

(Insert description of the scope of the project.)

3.2 Planned usage of the envisioned specification

Note: New airplane programs must be confirmed by manufacturer prior to completing this section.

New aircraft developments planned to use this specification yes ☐ no ☐

- Airbus: (aircraft & date)
- Boeing: (aircraft & date)
- Other: (manufacturer, aircraft & date)

Modification/retrofit requirement yes ☐ no ☐

- Specify: (aircraft & date)

Needed for airframe manufacturer or airline project yes ☐ no ☐

- Specify: (aircraft & date)

Mandate/regulatory requirement yes ☐ no ☐

Program and date: (program & date)

Is the activity defining/changing an infrastructure standard? yes ☐ no ☐

Specify (e.g., ARINC 429)

When is the ARINC standard required?
_____(month/year)_____

What is driving this date? _____(state reason)_____

Are 18 months (min) available for standardization work? yes ☐ no ☐

If NO please specify solution: _____

Are Patent(s) involved? yes ☐ no ☐

If YES please describe, identify patent holder: _____

3.3 Issues to be worked

(Describe the major issues to be addressed.)

4.0 Benefits

4.1 Basic benefits

Operational enhancements yes ☐ no ☐

For equipment standards:

(a) Is this a hardware characteristic? yes ☐ no ☐

(b) Is this a software characteristic? yes ☐ no ☐

(c) Interchangeable interface definition? yes ☐ no ☐

(d) Interchangeable function definition? yes ☐ no ☐

If not fully interchangeable, please explain: _____

Is this a software interface and protocol standard? yes ☐ no ☐

Specify: _____

Product offered by more than one supplier yes ☐ no ☐

Identify: (company name)

4.2 Specific project benefits (Describe overall project benefits.)

4.2.1 Benefits for Airlines

(Describe any benefits unique to the airline point of view.)

4.2.2 Benefits for Airframe Manufacturers

(Describe any benefits unique to the airframe manufacturer's point of view.)

4.2.3 Benefits for Avionics Equipment Suppliers

(Describe any benefits unique to the equipment supplier's point of view.)

5.0 Documents to be Produced and Date of Expected Result

Identify Project Papers expected to be completed per the table in the following section.

5.1 Meetings and Expected Document Completion

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

Activity	Mtgs	Mtg-Days (Total)	Expected Start Date	Expected Completion Date
Document a	*# of mtgs*	*# of mtg days*	*mm/yyyy*	*mm/yyyy*
Document b	*# of mtgs*	*# of mtg days*	*mm/yyyy*	*mm/yyyy*

Please note the number of meetings, the number of meeting days, and the frequency of web conferences to be supported by the IA Staff.

6.0 Comments

(Insert any other information deemed useful to the committee for managing this work.)

6.1 Expiration Date for the APIM

April/October 20XX

Completed forms should be submitted to the AEEC Executive Secretary.