

**Developer**

<b>COLLABORATORS</b>
----------------------

	<i>TITLE :</i> Developer		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 22, 2024	

<b>REVISION HISTORY</b>
-------------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>Developer</b>	<b>1</b>
1.1	Producer Developer Guide Contents . . . . .	1
1.2	Introduction . . . . .	1
1.3	Producer Structure . . . . .	2
1.4	Window Structures . . . . .	2
1.5	Menu Structures . . . . .	9
1.6	Image Structure . . . . .	10
1.7	Screen Structure . . . . .	10

# Chapter 1

## Developer

### 1.1 Producer Developer Guide Contents

The Producer Development Documentation

Introduction

Producer

Windows

Menus

Images

Screens

### 1.2 Introduction

This is the documentation associated with the shared producer library. You can use this to load Designer files and do what you like with the loaded data.

Include files exist containing the structure definitions. I believe that every structure has private data following what you see, so do not allocate or free any of this stuff yourself.

All you have to do is allocate a `ProducerNode` using `GetProducer` to be able to load data into it using `LoadDesignerData`. You then use the fields in the `ProducerNode` to get at the data.

Support is also given to the Standard Producer style windows. You can open one of these and update it using any of the listed functions.

Detection of Abort has also been improved and is very easy for you to do. Creation of `.cd` and `.ct` is done for you if you add the strings to the `LocaleList` in the `ProducerNode` using `AddLocaleString`.

I suggest you distribute the latest version of the library with your producer if you wish to distribute it. I would appreciate receiving a copy of your code if you use my library.

If you wish to import Designer Files into some other GUI creator that is fine by me.

---

## 1.3 Producer Structure

```

struct ProducerNode
{
    struct MinList          pn_WindowList;          List of WindowNodes.
    struct MinList          pn_MenuList;            List of MenuNodes.
    struct MinList          pn_ImageList;           List of ImageNodes.
    struct MinList          pn_ScreenList;          List of ScreenNodes.

    struct MinList          pn_LocaleList;          List of LocaleStrings.
    long                   pn_LocaleCount;          Number of localestrings
    char                    * pn_BaseName;          BaseName for locale.
    char                    * pn_GetString;          Getstring function name ←
        for locale.
    char                    * pn_BuiltInLanguage;    Locales built in ←
        language.
    long                   pn_LocaleVersion;         Locale version required.

    UBYTE                  pn_ProcedureOptions[50];  Not Used yet.
    UBYTE                  pn_CodeOptions[20];       See below
    UBYTE                  pn_OpenLibs[30];          Boolean stating ←
        whether to open a given library.
    long                   pn_VersionLibs[30];       Libraries are ordered ←
        as in the library window.
    UBYTE                  pn_AbortOnFailLibs[30];   Version and ←
        AbortOnFail use same ordering.
    char                    * pn_Includes;           Extra include files ←
        needed.
};

```

Code Options :

```

0    Comment Produced Code
1    Make WaitPointer Data
2    Create IDCMP Handlers
3    Make Library Code
4    Use __CHIP in C
5    OpenDiskFonts
6    Make Main Program
7    Write CD file
8    Write CT file
9    GTB Compatability
10   Alternate includes.
11   Open first screen
12   HSPascal V3.1 Units

```

## 1.4 Window Structures

```

struct WindowNode
{
    struct WindowNode      * wn_Succ;              Basic Window Structure
    struct WindowNode      * wn_Pred;
    struct MinList          wn_GadgetList;          List of Gadget Nodes.
    struct MinList          wn_TextList;            List of Text Nodes.
}

```

---

```

struct MinList      wn_ImageList;      List of SmallImage ↵
    nodes.
struct MinList      wn_BevelBoxList;    List of bevel boxes.
char                * wn_Label;         Window Label.
char                * wn_WinParams;     Extra params to be ↵
    placed in openwindow definition.
char                * wn_RendParams;     Extra params needed to ↵
    call rend window function.
struct TagItem      * wn_TagList;       Tags, see below.
struct MenuNode     * wn_Menu;          Menu associated with ↵
    window.
UBYTE               wn_LocaleOptions[6]; Localize which texts ↵
    options.
UBYTE               wn_CodeOptions[20];  Window function code ↵
    options.
UBYTE               wn_ExtraCodeOptions[20]; And some more of them.
UWORD               wn_Offx;            Offx as last used in ↵
    editor.
UWORD               wn_Offy;            Offy as above.
UWORD               wn_Fontx;           Fontx last used in ↵
    editor, needed for scaling.
UWORD               wn_Fonty;           Fonty as above.
long                wn_FirstID;         First Gadget ID.
};

```

LocaleOptions :

```

0    Gadget strings.
1    Text Strings.
2    WindowTitle.
3    ScreenTitle.

```

CodeOptions :

```

0    Check if already open.
1    If already open MoveToFront.
2    If already open activate.
3    Return Boolean for openwindow.
4    Only open if can make gadgets.
5    Use one gadget font.
6    Fail if already open.
7    Custom MsgPort.
8    Calculate border sizes.
9    Produce Gadget Array.
10   Make Rendwindow function public.
11   Slightly comment code.
12   Attach menu.
13   Create menustrip if not done.
14   Fail if cannot attach.
15   Free menu when closing.
16   Scale using screen font.
17   Make Workbench AppWindow.
18   Do not define some pointers

```

ExtraCodeOptions :

Private so far.

Tags :

Value	Type	Exists	Comment
-------	------	--------	---------

WA_Left	long	Always	
WA_Top	long	Always	
WA_Width	long	When not	WA_InnerWidth
WA_Height	long	When not	WA_InnerHeight
WA_InnerWidth	long	When not	WA_Width
WA_InnerHeight	long	When not	WA_Height
WA_Title	STRPTR	Always	
WA_ScreenTitle	STRPTR	Sometimes	Implement when exists
WA_MinWidth	long	Always	
WA_MinHeight	long	Always	
WA_MaxWidth	long	Always	
WA_MaxHeight	long	Always	
WA_SizeGadget	Boolean	Sometimes	Implement when exists
WA_SizeBRight	Boolean	Sometimes	Implement when exists
WA_SizeBBottom	Boolean	Sometimes	Implement when exists
WA_DragBar	Boolean	Sometimes	Implement when exists
WA_DepthGadget	Boolean	Sometimes	Implement when exists
WA_CloseGadget	Boolean	Sometimes	Implement when exists
WA_ReportMouse	Boolean	Sometimes	Implement when exists
WA_NoCareRefresh	Boolean	Sometimes	Implement when exists
WA_Borderless	Boolean	Sometimes	Implement when exists
WA_Backdrop	Boolean	Sometimes	Implement when exists
WA_GimmeZeroZero	Boolean	Sometimes	Implement when exists
WA_Activate	Boolean	Sometimes	Implement when exists
WA_RMBTrap	Boolean	Sometimes	Implement when exists
WA_Dummy + 0x030	Boolean	Sometimes	Implement when exists
WA_Dummy + 0x032	Boolean	Sometimes	Implement when exists
WA_Dummy + 0x037	Boolean	Sometimes	Implement when exists
WA_SimpleRefresh	Boolean	Sometimes	Implement when exists
WA_SmartRefresh	Boolean	Sometimes	Implement when exists
WA_AutoAdjust	Boolean	Sometimes	Implement when exists
WA_MenuHelp	Boolean	Sometimes	Implement when exists
WA_Zoom	UWORD *	Sometimes	Implement when exists
WA_MouseQueue	long	Sometimes	Implement when exists
WA_RptQueue	long	Sometimes	Implement when exists
WA_PubScreenFallBack	Boolean	Sometimes	Implement when exists
WA_PubScreen		Sometimes	If exists then take screen as ↵ parameter
WA_PubScreenName	STRPTR	Sometimes	Implement when exists
WA_CustomScreen		Sometimes	If exists then take screen as ↵ parameter
WA_IDCMP	long	Always	All that is required by gadtools ↵ gadgets as well as chosen IDCMP

```

struct SmallImageNode
{
    struct SmallImageNode * sin_Succ;      Image placed on window
    struct SmallImageNode * sin_Pred;
    struct ImageNode      * sin_Image;
    long                   sin_LeftEdge;
    long                   sin_TopEdge;
};

```

```

struct BevelBoxNode
{
    struct BevelBoxNode  * bb_Succ;

```

```

struct BevelBoxNode      * bb_Pred;
long                    bb_LeftEdge;
long                    bb_TopEdge;
long                    bb_Width;
long                    bb_Height;
UWORD                   bb_BevelType;    Type of bevel box, if unknown ←
    to your code draw type 0
};

```

I just draw them in rendwindow function.

Bevel Types at the moment are :

```

0 : Normal
1 : Recessed
2 : Double      Two boxes, one recessed inside the other.
3 : Sunk        Reverse of above.
4 : String      V39 kind.
5 : DropBox     V39 kind.

```

```

struct TextNode
{
    struct TextNode      * tn_Succ;
    struct TextNode      * tn_Pred;
    char                  * tn_Title;    Text
    long                  tn_LeftEdge;   Left coordinate
    long                  tn_TopEdge;    Top coordinate
    struct TextAttr       tn_Font;       Use if not screen font
    UBYTE                 tn_FrontPen;   Front pen colour
    UBYTE                 tn_BackPen;    Back pen colour
    UBYTE                 tn_DrawMode;   Drawmode
    UBYTE                 tn_ScreenFont; Use screen font boolean
};

```

These should be turned into intuitexts, see locale array to decide whether to localize

```

struct GadgetNode
{
    struct GadgetNode     * gn_Succ;
    struct GadgetNode     * gn_Pred;
    char                  * gn_Label;
    char                  * gn_Title;
    struct TagItem        * gn_TagList;
    long                  gn_Flags;
    long                  gn_LeftEdge;
    long                  gn_TopEdge;
    long                  gn_Width;
    long                  gn_Height;
    long                  gn_GadgetID;
    long                  gn_Kind;
    struct TextAttr       gn_Font;
};

```

Again all fields should be quite obvious.

Tags :



Value	Type	Exists	Comment
BUTTON_KIND			
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
CHECKBOX_KIND			
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
GTBX_Checked	Boolean	Sometimes	Implement if found
GT_TagBase+68 scale in V39+	Boolean	Sometimes	Implement if found, this is ←
CYCLE_KIND			
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
GTCY_Active	long	Sometimes	Implement if found
GTCY_Labels	STRPTR *	Always	Pointer to an array of STRPTR
INTEGER_KIND			
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
GTIN_MaxChars	long	Sometimes	Implement if found
STRINGA_Justification	STRPTR *	Sometimes	Implement if found
STRINGA_ReplaceMode	Boolean	Sometimes	Implement if found
STRINGA_ExitHelp	Boolean	Sometimes	Implement if found
GTST_EditHook	STRPTR	Sometimes	Insert in source if found
GA_TabCycle	Boolean	Sometimes	Implement if found
GA_Immediate	Boolean	Sometimes	Implement if found
LISTVIEW_KIND			
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
GTLV_Labels	List *	Sometimes	Implement if found
GTLV_ShowSelected	long	Sometimes	Either 0 or a pointer to another ←
gadget node, the predecessor in the gadget list			
GT_TagBase+83	STRPTR	Sometimes	EditHook string
GTLV_ScrollWidth	long	Sometimes	Implement if found
GTLV_Selected	long	Sometimes	Implement if found
LAYOUTA_Spacing	long	Sometimes	Implement if found
GTLV_ReadOnly	Boolean	Sometimes	Implement if found
MX_KIND			
GT_Underscore	long	Sometimes	Implement if found
GT_TagBase+69	Boolean	Sometimes	Scale in V39 +
GTMX_Spacing	long	Sometimes	Implement if found
GT_TagBase+71	long	Sometimes	Placetext in V39+
GTMX_Active	long	Sometimes	Implement if found
GTMX_Labels	STRPTR *	Always	Pointer to Array of STRPTR
MYBOOL_KIND			

GA_Image	ImageNode *	Sometimes	Render for gadget
GA_SelectRender	ImageNode *	Sometimes	SelectRender for gadget
GA_IntuiText	IntuiText *	Sometimes	Intuitext for gadget
GA_UserData	long	Always	Activation Flags

## MYOBJECT\_KIND

GA_UserData	List *	Always	List of MyTagItem
TAG_USER+47	STRPTR	Always	Class Name
TAG_USER+48	long	Always	Class Type
TAG_USER+49	long	Always	Object Type
TAG_USER+50	Boolean	Always	Scale
TAG_USER+51	Boolean	Always	Dispose

## NUMBER\_KIND

GTNM_Number	long	Sometimes	Implement if found
GTNM_Border	Boolean	Sometimes	Implement if found
GT_TagBase+72	long	Sometimes	Implement if found
GT_TagBase+73	long	Sometimes	Implement if found
GT_TagBase+74	long	Sometimes	Implement if found
GT_TagBase+85	long	Sometimes	Implement if found
GT_TagBase+76	long	Sometimes	Implement if found
GT_TagBase+75	long	Sometimes	Implement if found

## PALETTE\_KIND

GTPA_Depth	long	Sometimes	Implement if found
GTPA_Color	long	Sometimes	Implement if found
GTPA_ColorOffset	long	Sometimes	Implement if found
GTPA_IndicatorWidth	long	Sometimes	Implement if found
GTPA_IndicatorHeight	long	Sometimes	Implement if found
GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found

## SCROLLER\_KIND

GTSC_Top	long	SomeTimes	Implement if found
GTSC_Total	long	SomeTimes	Implement if found
GTSC_Visible	long	SomeTimes	Implement if found
PGA_Freedom	long	SomeTimes	Implement if found
GTSC_Arrows	long	SomeTimes	Implement if found
GA_Immediate	long	SomeTimes	Implement if found
GA_RelVerify	long	SomeTimes	Implement if found
GA_Disabled	long	SomeTimes	Implement if found
GT_Underscore	long	SomeTimes	Implement if found

## SLIDER\_KIND

GTSL_Min	long	SomeTimes	Implement if found
GTSL_Max	long	SomeTimes	Implement if found
GTSL_DispFunc	STRPTR	SomeTimes	Implement if found
GTSL_Level	long	SomeTimes	Implement if found
PGA_Freedom	long	SomeTimes	Implement if found
GTSL_LevelFormat	STRPTR	SomeTimes	Implement if found
GTSL_MaxLevelLen	long	SomeTimes	Implement if found

GTSL_LevelPlace	long	SomeTimes	Implement if found
GA_Immediate	long	SomeTimes	Implement if found
GA_RelVerify	long	SomeTimes	Implement if found
GA_Disabled	long	SomeTimes	Implement if found
GT_Underscore	long	SomeTimes	Implement if found

## STRING\_KIND

GT_Underscore	long	Sometimes	Implement if found
GA_Disabled	Boolean	Sometimes	Implement if found
GTST_MaxChars	long	Sometimes	Implement if found
STRINGA_Justification	STRPTR *	Sometimes	Implement if found
STRINGA_ReplaceMode	Boolean	Sometimes	Implement if found
STRINGA_ExitHelp	Boolean	Sometimes	Implement if found
GTST_EditHook	STRPTR	Sometimes	Insert in source if found
GA_TabCycle	Boolean	Sometimes	Implement if found
GA_Immediate	Boolean	Sometimes	Implement if found

## TEXT\_KIND

GTTX_Text	char *	Sometimes	Implement if found
GTTX_Border	Boolean	Sometimes	Implement if found
GTTX_CopyText	Boolean	Sometimes	Implement if found
GT_TagBase+72	long	Sometimes	Implement if found
GT_TagBase+73	long	Sometimes	Implement if found
GT_TagBase+74	long	Sometimes	Implement if found
GT_TagBase+85	long	Sometimes	Implement if found

## struct MyTag

```

{
    struct MyTag      * mt_Succ;
    struct MyTag      * mt_Pred;
    char              * mt_Label;      Text of tag
    long              mt_Value;      Value of tag, if -1 then insert ↵
        text.
    long              mt_BufferSize;  Size of buffer pointed to by ↵
        mt_Data, leave alone.
    UBYTE             * mt_Data;      Data associated with tag
    UWORD             mt_TagType;     Type of tag, see below.
};

```

## Object My Tag Types :

```

#define TagTypeLong      Data is a long value, not a pointer
#define TagTypeBoolean   Data is a Boolean value.
#define TagTypeString    Data is a STRPTR, possibly null.
#define TagTypeArrayByte Data is a pointer to an array of byte, ↵
    buffersize is length in bytes of buffer.
#define TagTypeArrayWord Data is a pointer to an array of word, ↵
    buffersize is length in bytes of buffer.
#define TagTypeArrayLong Data is a pointer to an array of long, ↵
    buffersize is length in bytes of buffer.
#define TagTypeArrayString Data is a pointer to an array of STRPTR, this is ↵
    NULL terminated.
#define TagTypeStringList Data is a pointer to a MinList structure, list ↵
    contains StringNodes.
#define TagTypeUser      Data is a STRPTR, possibly null.

```

---

```

#define TagTypeVisualInfo    Pass VisualInfo in tag.
#define TagTypeDrawInfo      Pass DrawInfo in tag.
#define TagTypeIntuiText      Data is a pointer to the first IntuiText in a ↵
    list, it contains a pointer to the next.
#define TagTypeImage         Data is a pointer to an ImageNode.
#define TagTypeImageData      Data is a pointer to an ImageNode.
#define TagTypeLeftCoord      Pass Gadget left coord, possibly scaled.
#define TagTypeTopCoord       Pass Gadget top coord, possibly scaled.
#define TagTypeWidth          Pass Gadget width, possibly scaled.
#define TagTypeHeight         Pass Gadget height, possibly scaled.
#define TagTypeGadgetID       Pass GadgetID
#define TagTypeFont           Pass TextAttr in use by gadgets currently.
#define TagTypeScreen         Pass pointer to current screen.
#define TagTypeGadget         Data is a pointer to another GadgetNode in this ↵
    WindowNode.
#define TagTypeUser2          Data is a STRPTR, possibly null.

```

## 1.5 Menu Structures

```

struct MenuNode
{
    struct MenuNode    * mn_Succ;
    struct MenuNode    * mn_Pred;
    struct MinList      mn_MenuList;      List of MenuTitleNode
    char                * mn_Label;       Menu Label
    struct TagItem      * mn_TagList;     Tags, see below
    UBYTE               mn_LocaleMenu;    Use locale for all strings
};

struct MenuTitleNode
{
    struct MenuTitleNode * mt_Succ;
    struct MenuTitleNode * mt_Pred;
    struct MinList      mt_ItemList;     List of MenuItemNode
    char                * mt_Text;       Text of title
    char                * mt_Label;      Label of title
    UBYTE               mt_Disabled;     Disable Title
};

struct MenuItemNode
{
    struct MenuItemNode * mi_Succ;
    struct MenuItemNode * mi_Pred;
    struct MinList      mi_SubItemList;  List of MenuSubItemNode
    char                * mi_Text;       Text of item
    char                * mi_Label;      Label of item
    struct ImageNode    * mi_Graphic;    Image node associated with ↵
    item
    char                mi_CommKey;      Key associated with item
    UBYTE               mi_Disabled;     Standard Menu Fields
    UBYTE               mi_Checkit;
    UBYTE               mi_MenuToggle;
    UBYTE               mi_Checked;
    UBYTE               mi_Barlabel;
    long                mi_Exclude;
}

```

```

};

struct MenuSubItemNode
{
    struct MenuSubItemNode * ms_Succ;
    struct MenuSubItemNode * ms_Pred;
    char * ms_Text;           Text of subitem
    char * ms_Label;         Label of subitem
    struct ImageNode * ms_Graphic; Image node associated with subitem ↔
    char ms_CommKey;         Key associated with subitem
    UBYTE ms_Disabled;       Standard Menu Fields
    UBYTE ms_Checkit;
    UBYTE ms_MenuToggle;
    UBYTE ms_Checked;
    UBYTE ms_Barlabel;
    long ms_Exclude;
};

```

All pretty clear if you read about gadtools menus.

Tags :

```

GTMN_FrontPen  default = 0
GTMN_TextAttr  default = none
GTMN_NewLook = GT_TagBase + 67  Always exists.

```

## 1.6 Image Structure

```

struct ImageNode
{
    struct ImageNode * in_Succ;
    struct ImageNode * in_Pred;
    char * in_Label;
    WORD in_Width;
    WORD in_Height;
    WORD in_Depth;
    UBYTE in_PlanePick;
    UBYTE in_PlaneOnOff;
    UBYTE * in_ImageData;
    long in_SizeAllocated;
    UBYTE * in_ColourMap;
    long in_MapSize;
};

```

Should be quite self explanatory, most fields are just copied into an Intuition Image structure. See the example for details on converting ImageData and ColourMap to source code.

## 1.7 Screen Structure

```

struct ScreenNode
{

```

---

```

    struct ScreenNode      * sn_Succ;
    struct ScreenNode      * sn_Pred;
    char                   * sn_Label;
    struct TagItem          * sn_TagList;
    UBYTE                  sn_LocaleTitle;
};

```

sn\_Label  
Screens Label inside the produced code.

sn\_TagList;  
Loads of information in this taglist, see below.

sn\_LocaleTitle  
Localize the title or not.

Tags :

Value	Type	Exists	Comment
SA_Left	long	Always	
SA_Top	long	Always	
SA_Width	long	Always	
SA_Height	long	Always	
SA_Depth	long	Always	
SA_OverScan	long	Always	
SA_Font	TextAttr	If not SA_SysFont	
SA_SysFont	0 or 1	If not SA_Font	
SA_Behind	Boolean	Sometimes	
SA_Quiet	Boolean	Sometimes	
SA_ShowTitle	Boolean	Sometimes	
SA_AutoScroll	Boolean	Sometimes	
SA_DisplayID	long	Always	
SA_FullPalette	Boolean	Sometimes	
SA_Title	STRPTR	Always	
SA_PubName	STRPTR	Sometimes	If tag exists then use.
SA_Type	long	Sometimes	CUSTOMSCREEN
SA_Pens	UWORD *	Always	
SA_ColorArray	UWORD *	Sometimes	
SA_ErrorCode		Sometimes	If tag exists then implement.
SA_Draggable	Boolean	Sometimes	
SA_Exclusive	Boolean	Sometimes	
SA_SharePens	Boolean	Sometimes	
SA_Interleaved	Boolean	Sometimes	
SA_LikeWorkbench	Boolean	Sometimes	
SA_BitMap	Boolean	Sometimes	If tag exists then implement, if ↩
true create bitmap.			
SA_PubSig			If exists then implement.