# DataType

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>DataType | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | January 13, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# DataType

## 1.1 DataType-related Classes Index (for AmigaTalk© 1998-2002):

Start reading here - the main node.

TagList (parent class is Array)

DataType (parent class is Object)

DataTypeSystem

DataTypesClassTags

DTSpecialInfoTags

DTMethodTags

DTFrameInfoTags

BasicDataTypeTags

DataTypeToolTags

DataTypeErrorTags

DTSystemTags

AnimationTags

PictureTags

SoundTags

TextTags

IDNumbers

Other Intuition-related Classes:

Intuition

IDCMPFlags

WindowTags

WindowFlags

ScreenTags

GadgetActivation

GadgetAttributes

GadToolsAttributes

GadgetMethodIDs

GadgetFlags

GadgetTypes

SpecialTags

MenuFlags

RequesterFlags

IconTags

IconTypeTags

WorkbenchTags

WorkbenchFlags

AppMsgTags

BOOPSI-related Classes:

ICClass

MethodIDs

ImageTags

BoopsiNames

## 1.2 DataType-related Classes for AmigaTalk© 1998-2002:

Described herein are the classes & their methods for manipulating
Amiga-DataType objects with AmigaTalk. Except where otherwise
noted, classes associated with DataTypes have Dictionary as a parent
class. Class TagList & DataType are the only classes that are not singleton classes.
intuition <- Intuition new.
dataTypeSystem <- DataTypeSystem new
Will create all of the Tag Dictionaries for your usage. There is an
example in TagList documentation.
SEE ALSO, InitializeCommands script
Most of the comments associated with the various Symbol keys came from
the C-Header files in the SAS-C Compiler that's been used to make
the AmigaTalk system.
WARNING: These classes cannot be idiot-proofed & flexible at the
same time, so know what you're doing before using them.
The class hierarchy is:
TagList (parent class is Array )
DataType (parent class is Object )
DataTypeSystem - focal point for the subclasses:
DataTypesClassTags

DTSpecialInfoTags

DTMethodTags

DTFrameInfoTags

BasicDataTypeTags

DataTypeToolTags

DataTypeErrorTags

DTSystemTags

AnimationTags

PictureTags

SoundTags

TextTags

IDNumbers

Other Intuition-related Classes:

Intuition - focal point for the subcalsses:

IDCMPFlags

WindowTags

WindowFlags

ScreenTags

GadgetActivation

GadgetAttributes

GadToolsAttributes

GadgetMethodIDs

GadgetFlags

GadgetTypes

SpecialTags

MenuFlags

RequesterFlags

IconTags

IconTypeTags

WorkbenchTags

WorkbenchFlags

AppMsgTags

BOOPSI-related Classes:

ICClass

MethodIDs

ImageTags

BoopsiNames

## 1.3 DataType Class:

Class DataType is a class that serves as a parent class for
all DataType-related classes that the user might generate.
newDTObject: dtName tags: tagArray
Create a new DataType Object with the given tags.
disposeDTObject
Delete a DataType Object from memory.
addDTObject: windowObj position: glistPos
Add a DataType Object to the given window & place it in the
glistPos slot.
removeDTObject: windowObj
Remove a DataType Object from the given window.
doAsyncLayout: layoutMsg
Perform an Asyhchronous layout (layouMsg) on the receiver.
doDTMethod: windowObj req: reqObj msg: message
Perform the given DataType Method (message).
getDTAttrs: tagArray
Retrieve the DataType attributes requested & place them in the
given tagArray Object , overwriting any old values found there.
(SEE ALSO, Array Class description )
getDTMethods
Retrieve an array of methods that the DataType object supports.
getDTString: stringID
Return a (error?) String for the given stringID Integer .
getDTTriggerMethods
Retrieve an array of DTMethod Objects that the DataType object
responds to.
examineFile: filename attrs: tagArray
Examine the data that the file points to & return a DataType
record that describes the data.
examineClip: clipHandle attrs: tagArray
Examine the data that the clipboard handle points to &
return a DataType record that describes the data.
printDTObject: windowObj req: reqObj prtObj: prtMsg
Tell the DataType Object to call the DTM_PRINT Method on a
separate process.
refreshDTObject: windowObj attrs: tagArray
Refresh a DataType Object.

releaseDTObject
Release a DataType struct obtained via ATObtainDataTypeA().
setDTAttrs: windowObj req: reqObj tags: tagArray
Set the attributes for a DataType Object.
translateDTErrorNum
Return a string for the IoErr() code number internally generated.
copyDTMethods: theArray include: inclusions exclude: exclusions
Clone and modify DTA_Methods array. nil returned on error.
copyDTTriggerMethods: methods include: incl exclude: excl
Clone and modify DTA_TriggerMethods array. nil returned on error.
obtainDomain: obj window: windowObj req: reqObj rport: rastPort
which: type domain: domain tags: attrTags
Obtain the min/nom/max domains of a dt object. This is equivalent
to the DoDTDomainA() function call in C. On success, the domain box
Object will be filled with the gadget's domain dimensions for this
particular GDOMAIN_#? id. nil is returned if there is an error.
drawDTObject: obj rport: rastPort start: point1 end: point2
h: htop v: vtop attrs: attrTags ! x y w h !
This method is used to draw a DataTypes object into a RastPort.
This method can be used for strip printing the object or
embedding it within a document. Returns true if successful.
findThisMethod: method in: methodsArrayObj
Find a specified method in methods array. Returns nil on error.
findToolNode: toolList attrs: attrTags
This method searches for a given tool in a list of tool nodes.
nil is returned if there is an error.
findTriggerMethod: dtnObj command: cmdStr method: methodNumber
This Method searches for a given trigger method in a given methods
array like that obtained from getDTTriggerMethods.
If one of the command or method args matches an array item, this
method returns a pointer to it. Returns nil on error.
freeDTMethods: methodsArray
Free methods array obtained by CopyDT#?Methods.
getDTTriggerMethodDataFlags: methodNumber
This method returns the kind of data which can be attached
to the stt_Data field in the dtTrigger method body.
The data type can be specified by or'ing the method id (within
STMF_METHOD_MASK value) with one of the STMD_#? identifiers:
STMD_VOID - stt_Data MUST be NULL

STMD_ULONG - stt_Data contains an unsigned long value
STMD_STRPTR - stt_Data is a string pointer
STMD_TAGLIST - stt_Data points to an array of struct TagItem's,
terminated with TAG_DONE
launchTool: toolObj project: projectString attrs: attrTags
This method launches an application with a specified project.
The application and it's launch mode and other attributes are
specified through the "Tool" structure.
INPUTS
tool - Pointer to a Tool structure. nil is a valid arg.
project - Name of the project to execute or nil .
attrs - Additional attributes.
TAGS
NP_Priority (BYTE) - sets the priority of the launched tool
Defaults to the current process's priority for
Shell and ARexx programs; Workbench applications
default to 0 except when overridden by the TOOLPRI tooltype.
NP_Synchronous (BOOL) - don't return until lauched application
process finishes. Defaults to false.
Returns false for failure, true otherwise.
lockDataType: dtObj
This method is used to lock a DataType structure obtained
by examineFile:attrs:, examineClip:attrs: or a datatypes
object (DTA_DataType attribute).
All calls to lockDataType, examineFile:attrs: or examineClip:attrs:
must match the same number of releaseDataType calls, otherwise
havoc will break out.
obtainDTDrawInfo: object attrs: attrTags
This method is used to prepare a DataTypes object for
drawing into a RastPort.
This method will send the DTM_OBTAINDRAWINFO method
to the object using the opSet message structure.
Returns nil on error, an Integer handle (for releaseDTDrawInfo:handle:)
otherwise.
releaseDTDrawInfo: anObject handle: aHandle
This method is used to release the information obtained
with obtainDTDrawInfo:attrs:
This method invokes the object's DTM_RELEASEDRAWINFO method
using the dtReleaseDrawInfo message structure.

saveDTObject: obj window: windowObj req: reqObj
file: filename mode: filemode flag: saveIconBool attrs: attrTags
This method saves the contents of an object into a file.
The method opens the named file and saves the object's contexts
into it (DTM_WRITE). Then it closes the file.
If the DTM_WRITE method returns success and saveIconBool is
true, the matching icon is saved.
If DTM_WRITE returns 0, the file will be deleted. Returns nil on
failure, an Integer otherwise.
startDragSelect: onObject
This method starts drag-selection by the user (marking).
This method replaces the old flag-fiddling method to
start drag-select.
The drag-select will only be started if the object supports
DTM_SELECT, is in a window or requester and no layout-process
is working on the object. If all conditions are good, it sets
the DTSIF_DRAGSELECT flag and returns true for success.
onObject is from newDTObject:tags:
cleanupDataTypes
Use this method only once, when you are finished with your DataType
Object. Using it more than once only slows the System down, no
other harm is done. If you don't use this method, datatypes.library
will remain open!

## 1.4 DataTypeSystem Class:

DataTypeSystem Class is a Singleton class that allows the user to
reference datatype-specific singleton classes in one spot.
The following classes are created by an instance of this class:
CLASS: DESCRIPTION:
========== ==========
DataTypesClassTags
DTSpecialInfoTags
DTMethodTags
DTFrameInfoTags
BasicDataTypeTags
DataTypeToolTags
DataTypeErrorTags
DTSystemTags

AnimationTags

PictureTags

SoundTags

TextTags

ImageTags

IDNumbers IFF Chunk ID numbers class.

Methods of interest to the user are:

new

Create the instance of DataTypeSystem.

getDTClassTag: key

Return the DTClassTag value associated with the Symbol key given.

getDTSpecialInfoTag: key

^ dtSpecialInfoTags at: key

getDTMethodTag: key

^ dtMethodTags at: key

getDTFrameInfoTag: key

^ dtFrameInfoTags at: key

getBasicDTTag: key

^ basicDTTags at: key

getDTToolTag: key

^ dtToolTags at: key

getDTErrorTag: key

^ dtErrorTags at: key

getDTSystemTag: key

^ dtSystemTags at: key

getDTAnimationTag: key

^ dtAnimTags at: key

getDTPictureTag: key

^ dtPictureTags at: key

getDTSoundTag: key

^ dtSoundTags at: key

getDTTextTag: key

^ dtTextTags at: key

getImageTag: key

^ imageTags at: key

getIDNumber: key

Return the IFF Chunk ID value associated with the Symbol key.

## 1.5 DataTypesClassTags Class:

DataTypesClassTags Class is a Singleton class that allows the user
to reference DataType class tags' hexadecimal values. This class
defines the following Symbol keys for usage in your DataType-related
classes:
#DTA_Dummy
Generic attributes:
#DTA_TextAttr Pointer to the default TextAttr to use for the
text within the object.
#DTA_TopVert Current top vertical unit.
#DTA_VisibleVert Number of visible vertical units.
#DTA_TotalVert Total number of vertical units.
#DTA_VertUnit Number of pixels per vertical unit.
#DTA_TopHoriz Current top horizontal unit.
#DTA_VisibleHoriz Number of visible horizontal units.
#DTA_TotalHoriz Total number of horizontal units.
#DTA_HorizUnit Number of pixels per horizontal unit.
#DTA_NodeName Name of the current element within the object.
#DTA_Title Title of the object.
#DTA_TriggerMethods Pointer to a NULL term'd array of trigger methods.
#DTA_Data Object specific data.
#DTA_TextFont Default font to use for text within the object.
#DTA_Methods Pointer to a ~0 terminated array of supported methods.
#DTA_PrinterStatus Printer error message.
#DTA_PrinterProc Pointer to the print process.
#DTA_LayoutProc Pointer to the layout process.
#DTA_Busy Used to turn the applications' busy pointer off/on.
#DTA_Sync Used to indicate that new information has been loaded into
an object. This is for models that cache the
DTA_TopVert-like tags
#DTA_BaseName The base name of the class.
#DTA_GroupID Group that the object must belong in.
#DTA_ErrorLevel Error level.
#DTA_ErrorNumber datatypes.library error number.
#DTA_ErrorString Argument for datatypes.library error.
#DTA_Conductor Specifies the name of the realtime.library conductor.
Defaults to 'Main'.
#DTA_ControlPanel Indicate whether a control panel should be embedded

within the object (in the animation datatype,
for example). Defaults to TRUE.
#DTA_Immediate (BOOL) Indicate whether the object should immediately
begin playing. Defaults to FALSE.
#DTA_Repeat (BOOL) Indicate that the object should repeat
playing. Defaults to FALSE.
#DTA_SourceAddress Address of a DTST_MEMORY source type object.
#DTA_SourceSize Size of a DTST_MEMORY source type object.
#DTA_Reserved Reserved tag; DO NOT USE (V44).
DTObject attributes:
#DTA_Name
#DTA_SourceType
#DTA_Handle
#DTA_DataType
#DTA_Domain
DON'T USE THE FOLLOWING EIGHT TAGS. Use the corresponding tags in
GadgetAttribute class:
#DTA_Left
#DTA_Top
#DTA_Width
#DTA_Height
#DTA_RelRight
#DTA_RelBottom
#DTA_RelWidth
#DTA_RelHeight
#DTA_ObjName
#DTA_ObjAuthor
#DTA_ObjAnnotation
#DTA_ObjCopyright
#DTA_ObjVersion
#DTA_ObjectID
#DTA_UserData
#DTA_FrameInfo
#DTA_SelectDomain
#DTA_TotalPVert
#DTA_TotalPHoriz
#DTA_NominalVert
#DTA_NominalHoriz
Printing attributes:

#DTA_DestCols Destination X width.

#DTA_DestRows Destination Y height.

#DTA_Special Option flags.

#DTA_RastPort RastPort to use when printing. (V40).

#DTA_ARexxPortName Pointer to base name for ARexx port (V40).

#DTST_RAM

#DTST_FILE

#DTST_CLIPBOARD

#DTST_HOTLINK

#DTST_MEMORY New for V44 of the AmigaOS.

The only method of interest to the user is:

new

Create an instance of the class.

## 1.6 DTSpecialInfoTags Class:

DataTypesClassTags Class is a Singleton class that allows the user
to reference DataType class tags' hexadecimal values. This class
defines the following Symbol keys:

#DTSIF_LAYOUT Object is in layout processing.

#DTSIF_NEWSIZE Object needs to be lain-out.

#DTSIF_DRAGGING

#DTSIF_DRAGSELECT

#DTSIF_HIGHLIGHT

#DTSIF_PRINTING Object is being printed.

#DTSIF_LAYOUTPROC Object is in layout process.

The only method of interest to the user is:

new

Create an instance of DTSpecialInfoTags class.

## 1.7 DTMethodTags Class:

DTMethodTags Class is a Singleton class that allows the user
to reference DataType Method tags' hexadecimal values. The
following Symbol keys are defined by this class:

#DTM_Dummy

#DTM_FRAMEBOX Inquire what environment an object requires.

#DTM_PROCLAYOUT Same as GM_LAYOUT except guaranteed to be on a process.

#DTM_ASYNCLAYOUT Layout that is occurring on a process.

#DTM_REMOVEDTOBJECT When a RemoveDTObject() is called.

#DTM_SELECT

#DTM_CLEARSELECTED

#DTM_COPY

#DTM_PRINT

#DTM_ABORTPRINT

#DTM_NEWMEMBER

#DTM_DISPOSEMEMBER

#DTM_GOTO

#DTM_TRIGGER

#DTM_OBTAINDRAWINFO

#DTM_DRAW

#DTM_RELEASEDRAWINFO

#DTM_WRITE

The only method of interest to the user is:

new

Create an instance of DTMethodTags class.

## 1.8 DTFrameInfoTags Class:

DTFrameInfoTags Class is a Singleton class that allows the user to reference DataType FrameInfo tags' hexadecimal values. This class defines the following Symbol keys:

#FIF_SCALABLE

#FIF_SCROLLABLE

#FIF_REMAPPABLE

The only method of interest to the user is:

new

Create an instance of DTFrameInfoTags class.

## 1.9 BasicDataTypeTags Class:

BasicDataTypeTags Class is a Singleton class that allows the user to reference Basic DataType tags' hexadecimal values. This class defines the following Symbol keys:

#DTF_TYPE_MASK

#DTF_BINARY

#DTF_ASCII

#DTF_IFF

#DTF_MISC

#DTF_CASE Set if case is important.

#DTF_SYSTEM1 Reserved for system use.

The only method of interest to the user is:

new

Create an instance of BasicDataTypeTags class.

## 1.10 DataTypeToolTags Class:

DataTypeToolTags Class is a Singleton class that allows the user to reference DataType Tool tags' hexadecimal values. This class defines the following Symbol keys:

For tn_Which:

#TW_INFO

#TW_BROWSE

#TW_EDIT

#TW_PRINT

#TW_MAIL

For tn_Flags:

#TF_LAUNCH_MASK

#TF_SHELL

#TF_WORKBENCH

#TF_RX

The only method of interest to the user is:

new

Create an instance of DataTypeToolTags class.

## 1.11 DataTypeErrorTags Class:

DataTypeErrorTags Class is a Singleton class that allows the user to reference DataType Error tags' hexadecimal values. This class defines the following Symbol keys:

Text ID's:

#DTERROR_UNKNOWN_DATATYPE

#DTERROR_COULDNT_SAVE

#DTERROR_COULDNT_OPEN

#DTERROR_COULDNT_SEND_MESSAGE

#DTERROR_COULDNT_OPEN_CLIPBOARD

#DTERROR_Reserved

#DTERROR_UNKNOWN_COMPRESSION

#DTERROR_NOT_ENOUGH_DATA

#DTERROR_INVALID_DATA

#DTERROR_NOT_AVAILABLE

#DTMSG_TYPE_OFFSET Offset for types.

The only method of interest to the user is:

new

Create an instance of DataTypeErrorTags class.

## 1.12 DTSystemTags Class:

DTSystemTags Class is a Singleton class that allows the user to reference DataType System tags' hexadecimal values. This class defines the following Symbol keys:

#STM_PAUSE

#STM_PLAY

#STM_CONTENTS

#STM_INDEX

#STM_RETRACE

#STM_BROWSE_PREV

#STM_BROWSE_NEXT

#STM_NEXT_FIELD

#STM_PREV_FIELD

#STM_ACTIVATE_FIELD

#STM_COMMAND

#STM_REWIND

#STM_FASTFORWARD

#STM_STOP

#STM_RESUME

#STM_LOCATE

The only method of interest to the user is:

new

Create an instance of DTSystemTags class.

## 1.13 AnimationTags Class:

AnimationTags Class is a Singleton class that allows the user to reference DataType Animation-class tags' hexadecimal values. This class defines the following Symbol keys:

Animation attributes:

#ADTA_Dummy

#ADTA_ModeID

#ADTA_KeyFrame Key frame (first frame) bitmap.

#ADTA_ColorRegisters

#ADTA_CRegs

#ADTA_GRegs

#ADTA_ColorTable

#ADTA_ColorTable2

#ADTA_Allocated

#ADTA_NumColors

#ADTA_NumAlloc

#ADTA_Remap Remap animation (defaults to TRUE).

#ADTA_Screen Screen to remap to.

#ADTA_Width

#ADTA_Height

#ADTA_Depth

#ADTA_Frames Number of frames in the animation.

#ADTA_Frame Current frame.

#ADTA_FramesPerSecond Frames per second.

#ADTA_FrameIncrement Amount to change frame by when fast forwarding or rewinding. Defaults to 10.

#ADTA_PreloadFrameCount Number of frames to preload; defaults to 10.

Sound attributes:

#ADTA_Sample

#ADTA_SampleLength

#ADTA_Period

#ADTA_Volume

#ADTA_Cycles

#ADTA_LeftSample

#ADTA_RightSample

#ADTA_SamplesPerSec

#ADTM_Dummy

#ADTM_LOADFRAME Used to load a frame of the animation.

#ADTM_UNLOADFRAME Used to unload a frame of the animation.

#ADTM_START Used to start the animation.

#ADTM_PAUSE Used to pause the animation (don't reset the timer).

#ADTM_STOP Used to stop the animation.

#ADTM_LOCATE Used to locate a frame in the anim' (as set by a slider).

#ADTM_LOADNEWFORMATFRAME Used to load a new format frame of the animation (V44).

#ADTM_UNLOADNEWFORMATFRAME Used to unload a new format frame of the animation (V44).

The only method of interest to the user is:

new

Create an instance of AnimationTags class.

## 1.14 PictureTags Class:

PictureTags Class is a Singleton class that allows the user
to reference Picture DataType tags' hexadecimal values. This
class defines the following Symbol keys:
Picture attributes:
#PDTA_ModeID Mode ID of the picture.
#PDTA_BitMapHeader Bitmap header information.
#PDTA_BitMap Pointer to a class-allocated bitmap, that will end
up being freed by picture.class when DisposeDTObject()
is called.
#PDTA_ColorRegisters Picture colour table.
#PDTA_CRegs Color table to use with SetRGB32CM().
#PDTA_GRegs Color table; this table is initialized during the layout
process and will contain the colours the picture will use
after remapping. If no remapping takes place, these
colors will match those in the PDTA_CRegs table.
#PDTA_ColorTable Shared pen table; this table is initialized during
the layout process while the picture is being remapped.
#PDTA_ColorTable2 Shared pen table; in most places this table will be
identical to the PDTA_ColorTable table. Some of the
colors in this table might match the original color
palette a little better than the colors picked for
the other table. The picture.datatype uses the two
tables during remapping, alternating for each pixel.
#PDTA_Allocated OBSOLETE; DO NOT USE.
#PDTA_NumColors Number of colors used by the picture.
#PDTA_NumAlloc Number of colors allocated by the picture.
#PDTA_Remap Remap the picture (BOOL); defaults to TRUE.
#PDTA_Screen Screen to remap to.
#PDTA_FreeSourceBitMap Free the source bitmap after remapping.
#PDTA_Grab Pointer to a Point structure.
#PDTA_DestBitMap Pointer to the destination (remapped) bitmap.

#PDTA_ClassBitMap Pointer to class-allocated bitmap, that will end
up being freed by the class after DisposeDTObject()
is called.
#PDTA_NumSparse Number of colors used for sparse remapping.
#PDTA_SparseTable Pointer to a table of pen numbers indicating
which colors should be used when remapping the image.
This array must contain as many entries as there
are colors specified with PDTA_NumSparse.
#PDTA_WhichPicture Index number of the picture to load. (V44).
#PDTA_GetNumPictures Get the number of pictures stored in the file. (V44).
#PDTA_MaxDitherPens Maximum number of colors to use for dithering. (V44).
#PDTA_DitherQuality Quality of the dithering algorithm to be used during
color quantization. (V44)
#PDTA_AllocatedPens Pointer to the allocated pen table. (V44).
#PDTANUMPICTURES_Unknown When querying the # of pictures stored in a
file, the following value denotes the # of
pictures is unknown.
#PDTA_SourceMode Set the sub datatype interface mode.
#PDTA_DestMode Set the app datatype interface mode.
#PDTA_UseFriendBitMap Allocates the resulting bitmap as a friend bitmap.
#PDTA_MaskPlane NULL or mask plane for use with BltMaskBitMapRastPort().
Interface modes:
#PMODE_V42 Compatibility mode.
#PMODE_V43 Extended mode.
#PDTM_Dummy
#PDTM_WRITEPIXELARRAY Transfer pixel data to the picture object in
the specified format.
#PDTM_READPIXELARRAY Transfer pixel data from the picture object
in the specified format.
Pixel formats:
#PBPAFMT_RGB 3 bytes/pixel (rgb).
#PBPAFMT_RGBA 4 bytes/pixel (rgb, alpha).
#PBPAFMT_ARGB 4 bytes/pixel (alpha, rgb).
#PBPAFMT_LUT8 1 byte/pixel (using a separate color map).
#PBPAFMT_GREY8 1 byte/pixel (0 is black, 255 is white).
Masking techniques:
#mskNone
#mskHasMask
#mskHasTransparentColor

#mskLasso

#mskHasAlpha

Compression techniques:

#cmpNone

#cmpByteRun1

#cmpByteRun2 NOTE: unused (V44)

#DTM_WRITEPIXELARRAY

#DTM_READPIXELARRAY

#DTM_CREATEPIXMAPDIR

#DTM_FIRSTPIXMAPDIR

#DTM_NEXTPIXMAPDIR

#DTM_PREVPIXMAPDIR

#DTM_BESTPIXMAPDIR

#PDTA_SourceMode

#PDTA_DestMode

#PDTA_PixelFormat

#PDTA_TransRemapPen

#PDTA_NumPixMapDir

#PDTA_UseFriendBitMap

#PDTA_AlphaChannel

#PDTA_MultiRemap

#PDTA_MaskPlane

PDTA_SourceMode, PDTA_DestMode:

#MODE_V42

#MODE_V43

The only method of interest to the user is:

new

Create an instance of PictureTags class.

## 1.15 SoundTags Class:

SoundTags Class is a Singleton class that allows the user
to reference Sound DataType tags' hexadecimal values. This
class defines the following Symbol keys:

Sound attributes:

#SDTA_Dummy

#SDTA_VoiceHeader

#SDTA_Sample Sample data.

#SDTA_SampleLength Length of the sample data in UBYTEs.

#SDTA_Period Period.

#SDTA_Volume Volume. Range from 0 to 64.

#SDTA_Cycles

#SDTA_SignalTask Task to signal when sound is complete or next buffer

is needed.

#SDTA_SignalBit Signal mask to use on completion or 0 to disable

NOTE: Due to a bug in sound.datatype V40 SDTA_SignalBit

was actually implemented as a signal mask as

opposed to a bit number. The documentation now

reflects this. If you intend to use a signal bit number

instead of the mask, use the new V44 tag

SDTA_SignalBitNumber below.

#SDTA_SignalBitMask

#SDTA_Continuous Playing a continuous stream of data. Defaults to FALSE.

#SDTA_SignalBitNumber Signal bit to use on completion or -1 to disable.

#SDTA_SamplesPerSec Samples per second.

#SDTA_ReplayPeriod Sample replay period.

Sample data:

#SDTA_LeftSample

#SDTA_RightSample

#SDTA_Pan Stereo panning.

#SDTA_FreeSampleData FreeVec all sample data upon OM_DISPOSE.

#SDTA_SyncSampleChange Wait for the current sample to be played back before

switching to the new sample data.

Data compression methods:

#CMP_NONE

#CMP_FIBDELTA

#Unity Unity = Fixed 1.0 = maximum volume.

Channel allocation:

#SAMPLETYPE_Left

#SAMPLETYPE_Right

#SAMPLETYPE_Stereo

The only method of interest to the user is:

new

Create an instance of SoundTags class.

## 1.16 TextTags Class:

TextTags Class is a Singleton class that allows the user
to reference Text DataType tags' hexadecimal values.
This class defines the following Symbol keys:
Text attributes:
#TDTA_Buffer
#TDTA_BufferLen
#TDTA_LineList
#TDTA_WordSelect
#TDTA_WordDelim
#TDTA_WordWrap
#LNF_LF Line Feed.
#LNF_LINK Segment is a link.
#LNF_OBJECT ln_Data is a pointer to an DataTypes object.
#LNF_SELECTED Object is selected.
The only method of interest to the user is:
new
Create an instance of TextTags class.

## 1.17 TagList Class:

TagList Class is mainly used for Amiga function calls that utilize
Tags. AmigaTalk converts TagLists internally into what the AmigaOS
expects for real Amiga tags. TagList parent class is Array .
Since TagList is an Array to AmigaTalk, you should realize that
you have to set Tags & Values in pairs. Example:

```
intuiTags <- Intuition new
myTags <- TagList new: 24
myTags setTag: (intuiTags getWindowTag: #WA_Left) index: 0
myTags setTagValue: (intuiTags getWindowTag: #WA_Left) value: 16
myTags setTag: (intuiTags getWindowTag: #WA_Top) index: 2
myTags setTagValue: (intuiTags getWindowTag: #WA_Top) value: 0
...
myTags setTag: (intuiTags specialTag: #TAG_DONE) index: 24
```

Until myTags is filled. Do NOT forget to terminate your TagList
with (intuiTags specialTag: #TAG_DONE)
Methods to use for the TagList class:
new: newSize

Create a new instance of the TagList Class with newSize elements.

getTagValue: theTag

Return the value associated with theTag.

setTagValue: theTag value: newTagValue

Set a given tag (theTag) to newTagValue.

addTagPair: newTag value: newTagValue

Add an entry pair to the TagList. NOTE: This is an expensive, memory fragmenter. You should create the size TagList you need or make a larger one if you're not sure how large to make it.

deleteTagPair: theTag

Remove theTag and it's associated value from the TagList.

setTag: newTag index: idx

Set a tag at the array index idx to newTag.

## 1.18 IFF ID Numbers class:

IDNumbers Class is a Singleton class that allows the user to reference IDNumbers which denote IFF Chunk headers.

This class defines the following Symbol keys:

Used by multiple types:

#ID_BODY "BODY"

Animation IDs:

#ID_ANIM "ANIM"

#ID_ANHD "ANHD"

#ID_DLTA "DLTA"

DataType IDs:

#ID_DTYP "DTYP"

#ID_DTHD "DTHD"

#GID_SYSTEM "syst" System file, such as; directory, executable, library, device, font, etc.

#GID_TEXT "text" Formatted or unformatted text.

#GID_DOCUMENT "docu" Formatted text with graphics or other DataTypes.

#GID_SOUND "soun" Sound.

#GID_INSTRUMENT "inst" Musical instruments used for musical scores.

#GID_MUSIC "musi" Musical score.

#GID_PICTURE "pict" Still picture.

#GID_ANIMATION "anim" Animated picture.

#GID_MOVIE "movi" Animation with audio track.

A code chunk contains an embedded executable that

can be loaded with InternalLoadSeg:

#ID_CODE "DTCD"

#ID_TOOL "DTTL"

#ID_TAGS "DTTG"

#ID_NAME "NAME"

For Picture DataTypes -- IFF types that may be in pictures:

#ID_ILBM "ILBM"

#ID_BMHD "BMHD"

#ID_CMAP "CMAP"

#ID_CRNG "CRNG"

#ID_GRAB "GRAB"

#ID_SPRT "SPRT"

#ID_DEST "DEST"

#ID_CAMG "CAMG"

For Text datatypes -- IFF types that may be text:

#ID_FTXT "FTXT"

#ID_CHRS "CHRS"

For Sound DataTypes:

#ID_8SVX "8SVX"

#ID_VHDR "VHDR"

#ID_CHAN "CHAN"

The only method of interest to the user is:

new

Create an instance of IDNumbers class.

## 1.19 Intuition Class:

Intuition Class is a Singleton class that allows the user to
reference intuition-specific singleton classes in one spot.

NOTE:

This Class is very large (consumes close to 20MB), so if memory
usage is an issue for you, then DO NOT instantiate one! Just
instantiate the Tags or Flags classes that you actually need.

You will also have to edit the AmigaTalk:C/InitializeCommands
startup script also, since the Intuition Class is instantiated by
default (as intuition).

The following classes are instantiated from this class:

GadgetAttributes

GadgetFlags

GadgetTypes

GadToolsAttributes

GadgetActivation

GadgetMethodIDs

IDCMPFlags

ScreenTags

WindowTags

WindowFlags

SpecialTags

ICClass BOOPSI-related.

MethodIDs BOOPSI-related.

BoopsiNames BOOPSI-related.

ImageTags

MenuFlags

ReqFlags

IconTags

IconTypeTags

WorkbenchTags

WorkbenchFlags

AppMsgTags

This means that you only have to add the following statement to

your AmigaTalk code to create all of the Intuition-related

Tag Dictionaries:

myIntuition <- Intuition new.

Methods of interest to the user are:

new

Create an instance of Intuition class.

getBoopsiClassName: key

Retrieve the boopsi Class string associated with the key.

getRequesterFlag: key

Retrieve the Requester Flag associated with the key.

getWorkbenchTag: key

Retrieve the Workbench Tag associated with the key.

getWorkbenchFlag: key

Retrieve the Workbench Flag associated with the key.

getAppMsgTag: key

Retrieve the AppMsg Tag associated with the key.

getMenuFlag: key

Retrieve the Menu Flag associated with the key.

getIconTag: key

Retrieve the Icon Tag associated with the key.

getIconTypeTag: key

Retrieve the Icon Type Tag associated with the key.

specialTag: key

Return the SpecialTag value associated with the Symbol key.

icClass: key

Return the ICClass (BOOPSI) value associated with the Symbol key.

methodIDs: key

Return the MethodIDs (BOOPSI) value associated with the Symbol key.

getGadgetAttr: key

Return the GadgetAttributes value associated with the Symbol key.

getGadToolAttr: key

Return the GadToolsAttributes value associated with the Symbol key.

getGadgetFlag: key

Return the GadgetFlags value associated with the Symbol key.

getGadgetType: key

Return the GadgetTypes value associated with the Symbol key.

getGadgetMethodID: key

Return the GadgetMethodIDs value associated with the Symbol key.

getGadgetActivation: key

Return the GadgetActivation value associated with the Symbol key.

getScreenTag: key

Return the ScreenTags value associated with the Symbol key.

getWindowTag: key

Return the WindowTags value associated with the Symbol key.

getIDCMPFlag: key

Return the IDCMPFlags value associated with the Symbol key.

getWindowFlag: key

Return the WindowFlags value associated with the Symbol key.