datatypes

COLLABORATORS			
	TITLE : datatypes		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 23, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

datatypes

1.1 datatypes.doc

```
--background--
--rexxhost--
AddDTObject()
DisposeDTObject()
DoAsyncLayout()
DoDTMethodA()
GetDTAttrsA()
GetDTMethods()
GetDTString()
GetDTTriggerMethods()
NewDTObjectA()
ObtainDataTypeA()
PrintDTObjectA()
RefreshDTObjectA()
ReleaseDataType()
RemoveDTObject()
SetDTAttrsA()
```

1.2 datatypes.library/--background--

```
PURPOSE
    The datatypes.library provides transparent data handling
    abilities to applications.  Application developers can register
    their data format with datatypes.library and provide a class
    library for handling their data within other applications.

OVERVIEW

    * Object Oriented

    datatypes.library implementation is object oriented, using the
    boopsi functions of Intuition.  Each data class is implemented
    as a shared system library.

    * Embedded Objects
```

datatypes.library provides the ability to embed different object types within an application. For example, an application can embed an picture object or even an AmigaGuide document browser within their application's window. Objects can also be embedded within other objects.

* Gadget-like

Embedded objects are actually boopsi gadgets. That means that input handling is done on Intuition's task. Time intensive operations, such as layout when the window size changes, are off-loaded to a sub-process. Printing, clipboard operations, file read/write are also off-loaded to a separate process on an as-needed basis.

* Trigger Methods

Sometimes it is necessary for an application to provide additional controls for navigating through an object. For example, with an AmigaGuide object it is necessary to have controls for "Contents", "Index", "Browse >", "Browse <". Each class implements a method that returns the trigger methods, and the appropriate labels, that a class supports.

* Format Conversion

As long as the objects are sub-classes of the same class, data from one format can be written out as another format. For example, it is possible to read in an ILBM file and write out a JPEG file, since both data types are sub-classes of PICTURE.

* Future Compatible

Each class implements a method that returns the supported methods within a class. This way an application can ask an object if it is capable of any particular method, such as DTM_WRITE for example.

* Data Type Detection

datatypes.library provides the ability to examine a file or the clipboard to determine what type of data it contains.

1.3 datatypes.library/--rexxhost--

HOST INTERFACE

datatypes.library provides an ARexx function host interface that enables ARexx programs to take advantage of the features of data types. The functions provided by the interface are directly related to the functions described herein, with the differences mostly being in the way they are called.

The function host library vector is located at offset -30 from the library. This is the value you provide to ARexx in the

```
    AddLib() function call.

FUNCTIONS
    ExamineDT(FILENAME/A,VARIABLENAME,STEM/S,VAR/S)

EXAMPLE
    /* datatypes.rexx */
    PARSE ARG fname
    OPTIONS RESULTS

    /* Load the datatypes.library as a function host */
    IF ~SHOW('L','datatypes.library') THEN
       CALL ADDLIB('datatypes.library',0,-30)

       IF fname="" THEN DO
          SAY "Usage:"
          SAY "  rx datatypes <filename>"
          EXIT
       END

       SAY 'var test'
       type = ExamineDT(fname,,VAR)
       SAY type

       SAY 'stem test'
       CALL ExamineDT(fname,dtstem.,STEM)

       SAY '      Disk Key:' dtstem.DiskKey
       SAY 'Dir Entry Type:' dtstem.DirEntryType
       SAY '     File Name:' dtstem.FileName
       SAY '    Protection:' dtstem.Protection
       SAY '    Entry Type:' dtstem.EntryType
       SAY '          Size:' dtstem.Size
       SAY '        Blocks:' dtstem.NumBlocks
       SAY '          Date:' dtstem.Date
       SAY '       Comment:' dtstem.Comment
       SAY '          Type:' dtstem.BaseType
       SAY '     File Type:' dtstem.DataType
       SAY '     Base Name:' dtstem.BaseName
    EXIT
```

1.4 datatypes.library/AddDTObject

```
NAME
   AddDTObject - Add an object to a window or requester.   (V39)

SYNOPSIS
   realposition = AddDTObject (window, requester, object, position);
   d0                          a0      a1         a2      d0

   LONG AddDTObject (struct Window *, struct Requester *,
                     Object *, LONG);

FUNCTION
   This function adds a single object to the gadget list of the given
```

```
   window or requester at the position in the list specified by the
   position argument.

   When the object is added to the gadget list, the object will
   receive a GM_LAYOUT message with the gpl_Initial field set to
   one.

INPUTS
   window - Pointer to the window.

   requester - Must be NULL.

   object - Pointer to an object as returned by NewDTObjectA().

   position - Integer position in the list for the new gadget.
       -1 to add to the end of the list.

RETURNS
   Returns the position of where the object was actually added.

SEE ALSO
   RemoveDTObject(), intuition.library/AddGList()
```

1.5 datatypes.library/DisposeDTObject

```
NAME
   DisposeDTObject - Delete a data type object.            (V39)

SYNOPSIS
   DisposeDTObject (o);
                    a0

   VOID DisposeDTObject (Object *);

FUNCTION
   This function is used to dispose of a data type object that was
   obtained with NewDTObjectA().

INPUTS
   o - Pointer to an object as returned by NewDTObjectA().
       NULL is a valid input.

SEE ALSO
   NewDTObjectA()
```

1.6 datatypes.library/DoAsyncLayout

```
NAME
   DoAsyncLayout - Call the DTM_ASYNCLAYOUT method on a separate process.
                                                                (V39)

SYNOPSIS
```

```
   retval = DoAsyncLayout (object, gpl);
   d0                      a0      a1

   ULONG DoAsyncLayout (Object *, struct gpLayout *);

FUNCTION
   This function is used to asyncronously perform the object's
   DTM_ASYNCLAYOUT method.  This is used to offload the layout method
   from input.device.

   The DTM_ASYNCLAYOUT method must exit when SIGBREAKF_CTRL_C signal
   is set.   This indicates that the data has become obsolete and
   the DTM_ASYNCLAYOUT method will be called again.

INPUTS
   object - Pointer to the data types object.
   gpl - Pointer to a gpLayout message.

RETURNS

SEE ALSO
```

1.7 datatypes.library/DoDTMethodA

```
NAME
   DoDTMethodA - Do a datatypes method.                    (V39)

SYNOPSIS
   retval = DoDTMethodA (o, win, req, msg);
   d0                    a0 a1   a2   a3

   ULONG DoDTMethodA (Object *, struct Window *,
                      struct Requester *, Msg);

   retval = DoDTMethod (o, win, req, data, ...);

   ULONG DoDTMethod (Object *, struct Window *,
                     struct Requester *, ULONG, ...);

FUNCTION
   This function is used to perform a datatypes method.

INPUTS
   o - Pointer to an object as returned by NewDTObjectA().

   win - Window that the object is attached to.

   req - Requester that the object is attached to.

   msg - The message to send to the object.

RETURNS
   Returns the value returned by the method.

SEE ALSO
```

```
   intuition.library/DoGadgetMethod()
```

1.8 datatypes.library/GetDTAttrsA

```
NAME
   GetDTAttrsA - Obtain attributes for an object.          (V39)

SYNOPSIS
   retval = GetDTAttrsA (o, attrs);
   d0                    a0  a2

   ULONG GetDTAttrsA (Object *, struct TagItem *);

   retval = GetDTAttrs (o, tag1, ...);

   ULONG GetDTAttrs (Object *, Tag tag1, ...);

FUNCTION
   This function is used to get the attributes of a data type
   object.

INPUTS
   o - Pointer to an object as returned by NewDTObjectA().

   attrs - Attributes to get, terminated with TAG_DONE.  The data
       element of each pair contains the address of the storage
       variable.

RETURNS
   retval - Contains the number of attributes the system was able
       to obtain.

SEE ALSO
   SetDTAttrsA(), intuition.library/GetAttr()
```

1.9 datatypes.library/GetDTMethods

```
NAME
   GetDTMethods - Obtain methods an object supports.       (V39)

SYNOPSIS
   methods = GetDTMethods (object);
   d0                      a0

   ULONG GetDTMethods (Object *);

FUNCTION
   This function is used to obtain a list of methods that an
   object supports.

INPUTS
   object - Pointer to an object as returned by NewDTObjectA().
```

```
RETURNS
   Returns a pointer to a ~0 terminated ULONG array.  This array
   is only valid until the object is disposed off.

SEE ALSO
   GetDTTriggerMethods()
```

1.10 datatypes.library/GetDTString

```
NAME
   GetDTString - Obtain a DataTypes string.                (V39)

SYNOPSIS
   str = GetDTString (id);
   d0                 d0

   STRPTR GetDTString (ULONG id);

FUNCTION
   This function is used to obtain a pointer to a localized
   DataTypes string.

INPUTS
   id - ID of the string to obtain.

RETURNS
   A pointer to a NULL terminated string.

SEE ALSO
```

1.11 datatypes.library/GetDTTriggerMethods

```
NAME
   GetDTTriggerMethods - Obtain trigger methods supported by an object
                                                                (V39)

SYNOPSIS
   methods = GetDTTriggerMethods (object);
   d0                              a0

   struct DTMethods *GetDTTriggerMethods (Object *);

FUNCTION
   This function is used to obtain a list of trigger methods that an
   object supports.  This is so that an application can provide
   the appropriate controls for an object.  For example, an AmigaGuide
   object needs controls for "Contents", "Index", "Retrace", "Browse <",
   and "Browse >",

INPUTS
   object - Pointer to an object as returned by NewDTObjectA().
```

```
RETURNS
   Returns a pointer to a NULL terminated DTMethod list.  This list is
   only valid until the object is disposed off.

EXAMPLE
   To call the method:

       DoMethod (object, DTM_TRIGGER, dtm[button]->dtm_Method);

SEE ALSO
   GetDTMethods()
```

1.12 datatypes.library/NewDTObjectA

```
NAME
   NewDTObjectA - Create an data type object.              (V39)

SYNOPSIS
   o = NewDTObjectA (name, attrs);
   d0                d0   a0

   Object *NewDTObjectA (APTR, struct TagItem *);

   o = NewDTObject (name, tag1, ...);

   Object *NewDTObject (APTR, Tag tag1, ...);

FUNCTION
   This is the method for creating datatype objects from
   'boopsi' classes.  Boopsi' stands for "basic object-oriented
   programming system for Intuition".)

   You further specify initial "create-time" attributes for the
   object via a TagItem list, and they are applied to the
   resulting datatype object that is returned.

INPUTS
   name - Name of the data source.  Usually an existing file name.

   attrs - Pointer to a taglist containing additional arguments.

TAGS
   DTA_SourceType - Specify the type of source data; such as coming
       from a file or clipboard (defaults to DTST_FILE).  If
       source type is clipboard, then the name field contains the
       numeric clipboard unit.

   DTA_Handle - Can optionally be used instead of the name field.
       Must be a valid FileHandle if DTA_SourceType is DTST_FILE.
       Must be a valid IFFHandle if DTA_SourceType is DTST_CLIPBOARD.

   DTA_DataType - Specify the class of data.  Data is a pointer to a
       valid DataType.  This is only used when attempting to create
       a new object that doesn't have any source data.
```

```
    DTA_GroupID - Specify that the object must be of this type, or
        NewDTObject() will fail with IoErr() of
        ERROR_OBJECT_WRONG_TYPE.

    GA_Left, GA_RelRight, GA_Top, GA_RelBottom, GA_Width, GA_RelWidth,
    GA_Height, GA_RelHeight - Specify the placement of the object
        within the destination window.

    GA_ID - Specify the object ID.

    GA_UserData - Specify the application specific data for the
        object.


RETURNS
    A boopsi object, which may be used in different contexts such
    as a gadget or image, and may be manipulated by generic functions.
    You eventually free the object using DisposeDTObject().

    A NULL return indicates failure.  Use IoErr() to get error value.
    Following is a summary of the error number used and there meaning
    as it relates to DataTypes.

    ERROR_REQUIRED_ARG_MISSING - Indicates that a required attribute
        wasn't passed in.
    ERROR_BAD_NUMBER - An invalid group ID was passed in.
    ERROR_OBJECT_WRONG_TYPE - Object data type doesn't match
        DTA_GroupID.
    ERROR_NO_FREE_STORE - Not enough memory.
    DTERROR_UNKNOWN_DATATYPE - Unable to open the class library
        associated with the data type.
    DTERROR_COULDNT_OPEN - Unable to open the data object.
    ERROR_NOT_IMPLEMENTED - Unknown handle type.

NOTES
    This function invokes the OM_NEW "method" for the class specified.

EXAMPLE
    STRPTR fileName = "s:startup-sequence"
    Object *o;

    /* Simplest use is to just open an existing file */
    if (o = NewDTObject ((APTR)fileName, NULL))
    {
        /* Free the object when we are done with it */
        DisposeDTObject (o);
    }

SEE ALSO
    AddDTObject(), DisposeDTObject(), RemoveDTObject(),
    intuition.library/NewObjectA()
```

1.13 datatypes.library/ObtainDataTypeA

```
NAME
   ObtainDataTypeA - Examines a handle and return its DataType.
                                                                  (V39)
SYNOPSIS
   dtn = ObtainDataTypeA (type, handle, attrs);
   d0                     d0    a0      a1

   struct DataType *ObtainDataTypeA (ULONG, APTR, struct TagItem *);

   dtn = ObtainDataType (type, handle, tag1, ...);

   struct DataType *ObtainDataType (ULONG, APTR, Tag tag1, ...);

FUNCTION
   This function examines the data that the handle points to,
   and returns a DataType record that describes the data.

INPUTS
   type - Type of handle.

   handle - Handle to examine.
       For DTST_FILE, handle must be BPTR lock.
       For DTST_CLIPBOARD, handle must be struct IFFHandle *.

   attrs - Additional attributes (currently none are defined).

NOTES
   The datatypes.library maintains a sorted list of all the
   DataType descriptors.  The descriptor can consist of a
   function, a data mask for the first 64 bytes of the data,
   and a name pattern.

   The sort order for the list is:

       Descriptors with a function and no mask or name pattern.
       Descriptors with a function and a mask or name pattern.
       Descriptors with no function and a mask or name pattern.

   Within each group, they are also sorted in descending priority
   and descending mask length.

RETURNS
   Success returns a pointer to a DataType.  You must call
   ReleasDataType() when you are done with the handle.

   A NULL return indicates failure.  Use IoErr() to get error value.
   Following is a summary of the error number used and there meaning
   as it relates to DataTypes.

   ERROR_NO_FREE_STORE - Not enough memory.
   ERROR_OBJECT_NOT_FOUND - Unable to open the data object.
   ERROR_NOT_IMPLEMENTED - Unknown handle type.

SEE ALSO
   ReleaseDataType()
```

1.14 datatypes.library/PrintDTObjectA

```
NAME
   PrintDTObjectA - Call the DTM_PRINT method on a separate process.
                                                                (V39)

SYNOPSIS
   retval = PrintDTObjectA (object, window, requester, msg);
   d0                       a0      a1      a2         a3

   ULONG PrintDTObjectA (Object *, struct Window *, struct Requester *,
                         struct dtPrint *);

   retval = PrintDTObject (object, window, requester, data, ...);

   ULONG PrintDTObject (Object *, struct Window *, struct Requester *,
                        ULONG, ...);

FUNCTION
   This function is used to asyncronously perform the object's DTM_PRINT
   method.

   Once the application has performed the PrintDTObjectA() function,
   it must not manipulate the printerIO union until it receives a
   IDCMP_IDCMPUPDATE message that contains the DTA_PrinterStatus
   tag.

   To abort a print, the application sends the DTM_ABORTPRINT method
   to the object.  This in turn signals the print process with a
   SIGBREAKF_CTRL_C.

INPUTS
   object - Pointer to the DataTypes object.
   window - Pointer to the window that the object has been added to.
   requester - Pointer to the requester that the object has been
       added to.

RETURNS
   Returns TRUE if successful, FALSE on error.

SEE ALSO
```

1.15 datatypes.library/RefreshDTObjectA

```
NAME
   RefreshDTObjectA - Refresh a datatypes object.          (V39)

SYNOPSIS
   RefreshDTObjectA (object, window, req, attrs)
                     a0      a1      a2   a3

   VOID RefreshDTObjectA (Object *, struct Window *,
                          struct Requester *, struct TagItem *);
```

```
   RefreshDTObject (object, window, req, tag1, ...);

   VOID RefreshDTObject (Object *, struct Window *,
                         struct Requester *, Tag tag1, ...);

FUNCTION
   Refreshes the specified object, by sending the GM_RENDER method to
   the object.

INPUTS
   object - Pointer to an object as returned by NewDTObjectA().

   window - Pointer to the window.

   req - Must be NULL.

   attrs - Additional attributes (currently none are defined).

SEE ALSO
   AddDTObject(), RemoveDTObject(), intuition.library/RefreshGList()
```

1.16 datatypes.library/ReleaseDataType

```
NAME
   ReleaseDataType - Release a DataType structure.         (V39)

SYNOPSIS
   ReleaseDataType (dtn);
                    a0

   VOID ReleaseDataType (struct DataType *);

FUNCTION
   This function is used to release a DataType structure obtained
   by ObtainDataTypeA().

INPUTS
   dtn - DataType structure returned by ObtainDataTypeA().  NULL
       is a valid input.

SEE ALSO
   ObtainDataTypeA()
```

1.17 datatypes.library/RemoveDTObject

```
NAME
   RemoveDTObject - Remove an object from a window.        (V39)

SYNOPSIS
   position = RemoveDTObject (window, object);
   d0                         a0      a1
```

```
   LONG RemoveDTObject (struct Window *, Object *);

FUNCTION
   Removes the object from the window's object list.  This will wait
   until the AsyncLayout process is complete.  The object will
   receive a DTM_REMOVEDTOBJECT message to inform the object it has
   been removed.

INPUTS
   window - Pointer to the window.

   object - Pointer to an object as returned by NewDTObjectA().

RETURNS
   Returns the ordinal position of the removed object.  If the
   object wasn't found in the appropriate list then a -1 is
   returned.

SEE ALSO
   AddDTObject(), intuition.library/RemoveGList()
```

1.18 datatypes.library/SetDTAttrsA

```
NAME
   SetDTAttrsA - Set attributes for an object.             (V39)

SYNOPSIS
   retval = SetDTAttrsA (o, win, req, attrs);
   d0                    a0  a1   a2   a3

   ULONG SetDTAttrsA (Object *, struct Window *, struct Requester *,
                      struct TagItem *);

   retval = SetDTAttrs (o, win, req, tag1, ...);

   ULONG SetDTAttrs (Object *, struct Window *, struct Requester *,
                     Tag tag1, ...);

FUNCTION
   This function is used to set the attributes of a data type
   object.

INPUTS
   o - Pointer to an object as returned by NewDTObjectA().

   win - Window that the object has been added to.

   attrs - Attributes to set, terminated with TAG_DONE.

TAGS
   see <datatypes/datatypesclass.h> for tags.

SEE ALSO
   GetDTAttrsA(), intuition.library/SetGadgetAttrsA()
```