VIRTEC VT Client User Guide

Version 2.4.0

Table of Contents

Important Notice!

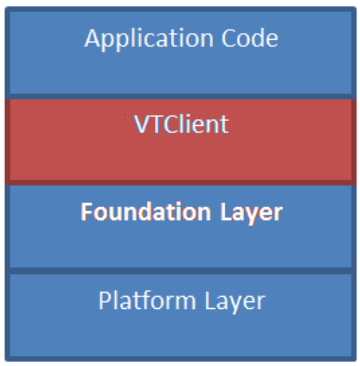
Note: VIRTEC Aux Control is currently in Beta. As such, this feature of the VT Client Library is not suitable for production use, and is currently released for development purposes only. With that said, Aux Control should be development-ready and a production-ready release is coming soon!

Overview

VIRTEC VT Client implements ISO 11783-6 (ISOBUS part 6) to allow interaction between the Virtual Terminal and an operator by connecting an electronic control unit from an implement to the VT. VT Client allows developers to establish connection and proper protocol for message transmission between VT and implement.

VT Auxiliary Control ("Aux Control") is a subsection of ISO 11783-6 (ISOBUS part 6), which along with Virtual Terminals defines the idea of standard operator controls (Auxiliary Inputs) that can be mapped to arbitrary implement functions (Auxiliary Functions).

As a part of the VT specification in ISOBUS, Aux Control is contained within the VT Client library of VIRTEC. This builds upon the VIRTEC Foundation library and the Platform layer to present the user with an interface to interact with a wide range of Aux Control-compatible devices.



VIRTEC Layers block diagram

Definitions

Aux Control
Auxiliary Control ("Aux" Control) is a section of ISOBUS part 6 that defines a set of generic operator controls that can be mapped to a given implement's functions. The machine operator can configure which Auxiliary Inputs are mapped to which of the implement's Auxiliary Functions, via the VT.

Aux Control Type 2
Also known as AUX-N on the AEF ISOBUS certification label. Contrasted with Aux Control Type 1 (AUX-O). Any controller reporting that it supports Aux Control version 3 or later *may not support* Type 1. AUX-O is not considered safe for use now that AUX-N is defined. VIRTEC supports AUX-N only.

Auxiliary Function ("Aux Function")
A function performed by an implement, which can be mapped to an Aux Input on the ISOBUS. An Aux Function may only ever be mapped to one Aux Input at a time.

Aux Function Type
Type of Aux Input (boolean latching, analog, etc. -- see [Aux Type 2 Function Types]). Also used with Aux Functions, to specify the type of input the function needs.

Auxiliary Input ("Aux Input")
An operator control that can be mapped to an Auxiliary Function on the ISOBUS. An Aux Input may have multiple Aux Functions mapped to it, which will be executed simultaneously. Aux inputs shall meet the operator control requirements specified in ISO 15077.

Aux Input Device
Also "Aux Input Unit". A device that incorporates one or more Aux Inputs, contained as a single ISOBUS Working Set.

Aux Input Model ID
The unique proprietary usigned 16-bit number that identifies a particular manufacturer's model and version of an Aux Input device. Newer incompatible versions of an Aux Input device must have a new unique Model ID. Valid range is 0x0-0xFFFE.

Virtual Terminal ("VT")
A virtual terminal consists of a graphical display and input functions, connected to an ISO 11783 network that provides the capability for an implement or a group of implements to interact with an operator. The VT provides the capability to display information and to retrieve data from an operator.

VT Client Guide

Setting up the VT Client

1. Run the *ConvertIOP.pl* tool (included with VIRTEC) to convert your object pool IOP files into C code that VIRTEC can access.
    1. Reference the `$USAGE` variable from the *ConvertIOP.pl* file for proper input parameters.
    2. Include the autogenerated header and C files into your VIRTEC project.
2. Create the VTClient_T object using the `MAKE_VTClient_T()` macro.
    1. Add the name of your [`Foundation_T`] object as the `foundation_ptr` parameter of your `MAKE_VTClient_T()` macro.
    2. Add the name of your array of VT_T objects as the `vt_array` parameter of your `MAKE_VTClient_T()` macro.
    3. If using Auxiliary Control fill in the `aux_function_list` and `aux_input_list` parameters appropriately or if not using Auxiliary Control fill the parameters in with `NULL`.
    4. Add the desired VTClient priority into the `priority` parameter of your `MAKE_VTClient_T()` macro.

Example

On the command prompt, enter:

```
perl ..\VTClient\ConvertIOP.pl -i=en:filepath\ObjectPool.iop -
g=filepath\ObjectPool.iop.h -o=MyAppObjectPool -l=MyApp -m=0x1234
```

And in the application's C code:

```
VTClient_T MyApp_VTClient = MAKE_VTClient_T(&MyApp_Foundation, MyApp_VTs,
NULL, NULL, MY_MUTEX_PRIORITY);
```

Using the VT Client

Connecting to a VT

1. Find the next active VT by using the VT_NextVT() function.
2. Connect to the VT by using the VT_Connect() function. This will cause the ISOBUS handshaking to start with the VT.
3. You can then send your object pool to the VT, or tell it to load one that it already has in its memory.
    1. Send your object pool directly by using the VT_SendObjectPool() function.
    2. If you've previously saved a version of your pool on the VT, then load it with the LoadVersion_Command() function.

Receiving events

Registering a VT Callback

1. Create a callback function to be associated to the VT callback structure.
2. Associate the callback to your app's VT client structure's VT callback list.
    - This is done by calling the appropriate `..._Register()` function.

Note that some messages are sent as soon as VIRTEC sees a VT on the bus in order to establish a connection with it. These messages are:

- Get Memory
- Get Number of Soft Keys
- Get Text Font Data
- Get Hardware
- Get Supported Wide Chars (VT v4 only)
- Get Window Mask Data (VT v4 only)
- Get Supported Objects (VT v4 only)

It's recommended to register for these messages at controller initialization, to make sure they're not missed.

(Alternatively, these metrics can be accessed through the `VT_T.Metrics` structure -- e.g., `MyApp_VT.Metrics.GetMemory`-- after the data is sent to the app; and definitely after the object pool is sent.)

NOTE: When using callback GetVersions_Callback_Register() the user is responsible for closing the pipe read handle member `Versions` via the `Pipe_CloseReadHandle()` function; e.g.:

```
static void My_GetVersions_Callback(VTClient_T *vt_client, const VT_T *vt,
GetVersions_Response_T *response)
{
   ...
```

```
    // done with the "response" data
    Pipe_CloseReadHandle(response->Versions);

    ...
}
```

The list of all callback `_Register()` functions available is the following:

- PointingEvent_Response_Callback_Register()
- VtControlAudioSignalTermination_Callback_Register()
- GetMemory_Response_Callback_Register()
- GetNumberOfSoftKeys_Callback_Register()
- GetTextFontData_Callback_Register()
- GetHardware_Callback_Register()
- GetSupportedWideChars_Callback_Register()
- GetWindowMaskData_Callback_Register()
- GetSupportedObjects_Callback_Register()
- GetVersions_Callback_Register()
- StoreVersion_Callback_Register()
- LoadVersion_Callback_Register()
- DeleteVersion_Callback_Register()
- ControlAudioSignal_Callback_Register()
- SetAudioVolume_Callback_Register()
- LockUnlockMask_Callback_Register()
- ExecuteMacro_Callback_Register()
- ChangeObjectLabel_Callback_Register()

Example:

Declare and define the callback:

```
static void GetMemory_Response_Function(VTClient_T *vt_client, const VT_T
*vt, const GetMemory_Response_T *cb_data)
{
    ...
}
```

Next, inside your app's init function, register the callback function using the appropriate `Register()` function:

```
void MyApp_Init(void)
{
  ...

  // Register VT Callback.
  GetMemory_Response_Callback_Register(&MyApp_VTClient,
GetMemory_Response_Function);
```

```
  ...
}
```

Finally, inside your app's task function, the following logic avoids blasting the bus with requests:

```
void MyApp_Task(void)
{
  ...
  if(send_getmemory_to_receive_getmemoryresponse_callback == TRUE)
  {
     // setting the "memory required" to 0 tells the VT to give us the
version of the standard it is built for (VTv2, VTv3, etc.)
     if(GetMemory_Message(&MyApp_VTClient, &MyApp_VT, NULL, 0))
     {
        send_getmemory_to_receive_getmemoryresponse_callback = FALSE;
     }
  }

  ...
}
```

Registering an Object Pool Callback

1. Create a callback function to be associated to the callback structure.
2. Create an object pool callback structure corresponding to the callback that is desired by using the correct MAKE macro.
    1. If a soft key activation callback is desired then a `SoftKeyActivation_Callback_T` structure needs to be made by using the `MAKE_SoftKeyActivation_Callback_T` macro.
    2. Place `EVERY_OBJECT_ID` in place of the `object_id` parameter of the MAKE macro if you want the callback function to be called regardless of Object ID.
3. Initialize the object pool by calling `ObjectPool_Init()`.
4. Register the callback by using the corresponding callback register function.
    1. If registering a `SoftKeyActivation_Callback_T` then use the corresponding register function `SoftKeyActivation_Register()`.

NOTE: When using callback `VtChangeStringValue_Callback_T` you are responsible for closing the `Pipe_ReadHandle_T` via the `Pipe_CloseReadHandle()` function.

Example

```
static void SoftKeyActivation_Function(const SoftKeyActivation_T *cb_data)
{
   ...
}

...
```

```
SoftKeyActivation_Callback_T SoftKeyActivation_Callback =
MAKE_SoftKeyActivation_Callback_T(SoftKeyActivation_Function, ObjectId_5000);

...

void Demo_Init(void)
{
   // Init the object pool
   ObjectPool_Init(&MyApp_ObjectPool);

   // Register Soft Key Activation callback
   (void)SoftKeyActivation_Register(&MyApp_ObjectPool,
&SoftKeyActivation_Callback);
}
```

Sending a Command

For object-related commands, access the VTv4_T structure. The `VTv4_T` structure is already defined by the library and can be accessed by calling `VTv4`.

1. Match the object type and select the desired command to be sent.
2. If no callback is desired enter `NULL` in place of the callback.

The `VTv4` structure also automatically scales any commands sent for a particular object, and handles the Attribute ID for any object attributes that need to be changed. There are low-level commands (see the next paragraph) that can also be called to do this sort of thing, but there the user has to supply all of the scaling and Attribute IDs themselves.

For non-object-related commands, there is usually a function associated with the message that needs to be sent to the VT. I.e., GetMemory_Message() can be called to send the Get Memory command to the VT.

Example

For object-related commands:

```
static uint16_t MeterSetting = 0;

...

MeterSetting++;

(void)VTv4.NumberVariable.ChangeNumericValue(&MyApp_VTClient, MyApp_VT, NULL,
NumberVariable_ObjectId, MeterSetting);

...
```

For non-object-related commands:

```
// Sends out the Get Memory Message to the VT.
(void)GetMemory_Message(&MyApp_VTClient, &MyApp_VT, NULL, 0);

// Another way to send out the Change Numeric Value command for a given
number variable
(void)ChangeNumericValue_Command((&MyApp_VTClient, &MyApp_VT, NULL,
NumberVariable_ObjectId, MeterSetting);
```

Other things you can do

Deleting your object pool from the VT's ROM

You can delete your working set's object pool from the VT's non-volatile memory by using the DeleteVersion_Command() function.

Disconnecting from a VT

Gracefully disconnect from the VT by using the VT_Disconnect() function. A graceful disconnect is the standard way of making sure the VT doesn't warn, alert, or complain to the operator that communication with your working set has been lost.

Example

```
bool_t disconnecting = VT_Disconnect(&MyApp_VTClient, MyApp_VT);
```

State Machine Example

Here is an example that incoporates a state machine to determine which of the above steps needs to be performed.

```
typedef enum AppState_E
{
   WAIT_VT,
   CONNECT_VT,
   DELETE_VERSIONS,
   SEND_OP,
   SEND_END_OP,
   OPERATOR_INTERACTION,
   DELETE_OP,
   DISCONNECT_VT,
   APP_IDLE
} AppState_T;

AppState_T MyApp_State;

...

void MyApp_Init(void)
{
    MyApp_State = WAIT_VT;
}
```

```
...

void MyApp_Task(void)
{
   switch (MyApp_State)
   {
   case WAIT_VT:
      if(VT_NextVT(&MyApp_VTClient, &MyApp_VT))
      {
         MyApp_State = CONNECT_VT;
      }
      break;
   case CONNECT_VT:
      if(VT_Connect(&MyApp_VTClient, MyApp_VT))
      {
         MyApp_State = DELETE_VERSIONS;
      }
      break;
   case DELETE_VERSIONS:
      // Deletes the last known version from the VT. Great for when you're
testing your
      // object pool and you want it to load the new one every time (without
having
      // to constantly change the version number).
      if(DeleteVersion_Command(&MyApp_VTClient, MyApp_VT, NULL, "       "))
      {
         MyApp_State = SEND_OP;
      }
      break;
   case SEND_OP:
      if(VT_SendObjectPool(&MyApp_VTClient, MyApp_VT, &MyApp_ObjectPool))
      {
         MyApp_State = SEND_END_OP;
      }
      break;
   case SEND_END_OP:
      if(MyApp_VT->ObjectPool.State == VT_OP_OPERATOR_INTERACTION)
      {
         MyApp_State = OPERATOR_INTERACTION;
      }
      else if(MyApp_VT->ObjectPool.State == VT_OP_IDLE)
      {
         MyApp_State = DELETE_OP;
      }
      break;
   case OPERATOR_INTERACTION:
      if(SoftwareTimer_Get(&MyApp_VT->Status.Timer) == TIMER_EXPIRED)
      {
         MyApp_State = WAIT_VT;
```

```
        }
        else if(MyApp_VT->ObjectPool.State == VT_OP_IDLE)
        {
            MyApp_State = WAIT_VT;
        }
        break;
    case DELETE_OP:
        if(VT_Disconnect(&MyApp_VTClient, MyApp_VT))
        {
            MyApp_State = WAIT_VT;
        }
        break;
    case DISCONNECT_VT:
        if(VT_Disconnect(&MyApp_VTClient, MyApp_VT))
        {
            MyApp_State = APP_IDLE;
        }
        break;
    case APP_IDLE:
    default:
        break;
    }
}
```

Aux Control Guide

Aux Control Initial Setup

Setting up Aux Control

Prior to Aux Control working, #defines must be added by the user in the platform.h file to enable the Aux Control features. The user must add the following #define that correlates to which functionality of Aux Control being used. 1. #define AUX_FUNCTION 1. Include this #define if using the Aux Function functionality from Aux Control 1. #define AUX_INPUT 1. Include this #define if using the Aux Input functionality from Aux Control

The user is responsible for loading the preferred assignments from non volatile memory and responsible for saving the preferred assignments to non volatile memory. The library will alert the user, through a callback, when the preferred assignments have been updated.

1. Create a PreferredAssignments_Updated_Callback_T object and initialize it with the [MAKE_PreferredAssignments_Updated_Callback_T] macro.
    1. Supply a callback to associate with the PreferredAssignments_Updated_Callback_T object.
2. Register the PreferredAssignments_Updated_Callback_T by calling VTClient_PreferredAssignments_Updated_Callback_Register().

3. In the callback the user is responsible for opening a pipe by calling the [Pipes_OpenFromCollection()] function and saving the preferred assignment data to non volatile memory by calling the VTClient_PreferredAssignments_Get() function.
    1. The function VTClient_PreferredAssignments_GetSize() is available to determine the size of pipe that needs to be opened.

    2. The user must close the read and write handles of the pipe after saving the preferred assignment data by calling the [Pipe_CloseReadHandle()] and [Pipe_CloseWriteHandle()] functions.
4. The user is also responsible for opening a pipe, at startup, and sending the preferred assignment data to the library from non volatile memory by calling the VTClient_PreferredAssignments_Set() function.
    1. The function VTClient_PreferredAssignments_GetSize() is available to determine the size of pipe that needs to be opened.
    2. The user must close the read and write handles of the pipe after saving the preferred assignment data by calling the [Pipe_CloseReadHandle()] and [Pipe_CloseWriteHandle()] functions.
    3. If there is no preferred assignment data stored in non volatile memory the library will default to no preferred assignments.

Example

```
// Create Callback Function
static void StorePreferredAssignments_Function(void)
{
   bool_t get = FALSE;
   Pipe_WriteHandle_T vtclient_handle;
   Pipe_ReadHandle_T nonvolatile_handle;

   // Open a pipe to pipe preferred assignment data into non volatile memory
   if (Pipes_OpenFromCollection(MYApp_VTClient.Foundation->Transport.RxPipes,
&vtclient_handle, &nonvolatile_handle,
VTClient_PreferredAssignments_GetSize(&MyApp_VTClient)))
   {
      // Writing data from the library into the write handle
      get = VTClient_PreferredAssignments_Get(&MyApp_VTClient,
vtclient_handle);

      // Let the non volatile memory read the preferred assignment data
      Pipe_CopyData(nonvolatile_handle, MyNonVolatilePreferredAssignmentData,
VTClient_PreferredAssignments_GetSize(&MyApp_VTClient));

      // Close pipe read/write handlers
      Pipe_CloseReadHandle(&nonvolatile_handle);
      Pipe_CloseWriteHandle(&vtclient_handle);
   }
}
```

```
// Initialize PreferredAssignments_Updated_Callback_T
static PreferredAssignments_Updated_Callback_T
StorePreferredAssignments_Callback =
MAKE_PreferredAssignments_Updated_Callback_T(StorePreferredAssignments_Functi
on);

...

void MyApp_Init(void)
{
   bool_t registered = FALSE;
   bool_t set = FALSE;
   Pipe_WriteHandle_T nonvolatile_handle;
   Pipe_ReadHandle_T vtclient_handle;

   // Register store preferred assignment callback
   registered =
VTClient_PreferredAssignments_Updated_Callback_Register(&MyApp_VTClient,
&StorePreferredAssignments_Callback);

   // Open a pipe to pipe preferred assignment data from non volatile memory
   if (Pipes_OpenFromCollection(MyApp_VTClient.Foundation->Transport.RxPipes,
&nonvolatile_handle, &vtclient_handle,
VTClient_PreferredAssignments_GetSize(&MyApp_VTClient)))
   {
      // Writing data from non volatile into the write handle
      Pipe_Insert(nonvolatile_handle, MyNonVolatilePreferredAssignmentData,
VTClient_PreferredAssignments_GetSize(&MyApp_VTClient));

      // Let the library read the preferred assignment data
      set = VTClient_PreferredAssignments_Set(&MyApp_VTClient,
vtclient_handle);

      // Close pipe read/write handlers
      Pipe_CloseReadHandle(&vtclient_handle);
      Pipe_CloseWriteHandle(&nonvolatile_handle);
   }
}
```

Aux Input Guide

Setting up Aux Inputs

1. Prior to proceeding ensure that the steps from Aux Control Setup have been performed.
2. Make sure that your copy of the VT Client includes the VIRTEC Aux Control feature. If you don't yet have this feature, contact [sales@distek.com].
3. Add one or more Aux Input objects to your object pool.

4. Run the *ConvertIOP.pl* tool (included with VIRTEC) to convert your object pool IOP files into C code that VIRTEC can access.
    1. Make sure to use the `-m <model-id>` flag, where `<model-id>` is your unique Aux Input Model ID.
    2. Include the autogenerated header and C files into your VIRTEC project.
5. Add the name of your AuxiliaryInputList_T object as the `aux_input_list` parameter of your `MAKE_VTClient_T()` macro. This object will come from the autogenerated C code from *ConvertIOP.pl*. Insert `NULL` in the `aux_function_list` parameter, if not using Aux Functions in your project.

Example

```
VTClient_T MyApp_VTClient = MAKE_VTClient_T(&MyApp_Foundation, MyApp_VTs,
NULL, &MyApp_AuxiliaryInputArray, MY_MUTEX_PRIORITY);
```

Using Aux Inputs

1. Call one of the Aux Input API functions to change the value of the given Aux Input on the ISOBUS. Pass a pointer to the Aux Input's AuxiliaryInput_T object -- these objects can be found in an array in the `VTClient_T` object.

Example

```
my_new_analog_value += 10;

AuxInput_Analog(&MyApp_VTClient.AuxiliaryInputList->AuxiliaryInputArray[4],
my_new_analog_value);
```

Aux Function Guide

Setting up Aux Functions

1. Prior to proceeding ensure that the steps from Auxiliary Control Setup have been performed.
2. Make sure that your copy of the VT Client includes the VIRTEC Aux Control feature. If you don't yet have this feature, contact [sales@distek.com].
3. Add one or more Aux Function objects to your object pool.
4. Run the *ConvertIOP.pl* tool (included with VIRTEC) to convert your object pool IOP files into C code that VIRTEC can access.
    1. Include the autogenerated header and C files into your VIRTEC project.
5. Add the name of your AuxiliaryFunctionList_T object as the `aux_function_ptr` parameter of your `MAKE_VTClient_T()` macro. This object will come from the autogenerated C code from *ConvertIOP.pl*. Insert `NULL` in the `aux_input_ptr` parameter, if not using Aux Inputs in your project.
6. For each Auxiliary Function you are supporting, create an AuxiliaryFunction_Callback_T object and initialize it with the MAKE_AuxFunction_Callback_T() macro. If one universal callback is desired for all

Auxiliary Functions register the callback with the object ID `EVERY_OBJECT_ID`. Both the universal and individual callbacks can be used together.

1. Supply callback functions for the assignment, maintenance and/or status messages as desired.
2. Supply `NULL` for any messages where you don't desire a callback function.
3. See the `AuxiliaryFunction_Callback_T` definition for details on the callback function pointers.

7. Register the `AuxiliaryFunction_Callback_T` with a given Aux Function, by calling AuxiliaryFunction_Callback_Register().

Example

Here's an example with the individual callbacks per Aux Function object:

```
VTClient_T MyApp_VTClient = MAKE_VTClient_T(&MyApp_Foundation, MyApp_VTs,
&MyApp_AuxiliaryFunctionArray, NULL, MY_MUTEX_PRIORITY);

static void AuxFunction2_29000_Status_Callback(const
AuxiliaryInputType2Status_Message_T *cb_data)
{
   ...
}

...

// Etc. for ...29000_Assignment_Callback and ...29000_Maintenance_Callback,
if desired.

...

AuxiliaryFunction_Callback_T AuxFunction2_29000_Callback =
MAKE_AuxFunction_Callback_T(AuxFunction2_29000_Assignment_Callback,
AuxFunction2_29000_Maintenance_Callback, AuxFunction2_29000_Status_Callback);

void MyApp_Init(void)
{
   ...
   bool_t registered = AuxiliaryFunction_Callback_Register(&MyApp_VTClient,
AuxFunction2_29000, &AuxFunction2_29000_Callback);
   ...
}
```

Here's another example with a universal Aux Function callback, that is called for all Aux Function objects. Note that both callback types (individual and universal) *can* exist together, though circumstances normally only require one or the other.

```
VTClient_T MyApp_VTClient = MAKE_VTClient_T(&MyApp_Foundation, MyApp_VTs,
&MyApp_AuxiliaryFunctionArray, NULL, MY_MUTEX_PRIORITY);
```

```
static void AuxFunction2_Universal_Status_Callback(const
AuxiliaryInputType2Status_Message_T *cb_data)
{
   switch (cb_data->AuxInputObjectID)
   {
   case(AuxFunction2_29000) :
      ...
   case(AuxFunction2_29001) :
      ...
   default:
      break;
   }
}

...

// Etc. for ...Universal_Assignment_Callback and
...Universal_Maintenance_Callback, if desired.

...

AuxiliaryFunction_Callback_T AuxFunction2_Universal_Callback =
MAKE_AuxFunction_Callback_T(AuxFunction2_Universal_Assignment_Callback,
AuxFunction2_Universal_Maintenance_Callback,
AuxFunction2_Universal_Status_Callback);

void MyApp_Init(void)
{
   ...
   bool_t registered = AuxiliaryFunction_Callback_Register(&MyApp_VTClient,
EVERY_OBJECT_ID, &AuxFunction2_Universal_Callback);
   ...
}
```

Using Aux Functions

1. Get the input values from the assigned Aux Input via the (*StatusFunction)() callback, which was assigned via MAKE_AuxFunction_Callback_T().
    1. The input values are in the passed-in `AuxiliaryInputType2Status_Message_T *cb_data` structure pointer.

Optionally, the user can monitor when assignments are added to and removed from a given Aux Function, via the (*AssignmentFunction)() callback. Users can also monitor the maintenance messages from the assigned Aux Input device via the (*MaintenanceFunction)() callback.

API Reference

VT Client API Reference

This section specifies all of the function calls, structures, and macros that make up the VIRTEC VTClient user interface. For details on any structures, objects, or functions that may be missing here, please see Annexes.h.

Data Types

AlarmPriority_T: uint8_t
AcousticSignal_T: uint8_t
ArchedBarGraphOptions_T: uint8_t
AttributeID_T: uint8_t
AttributeValue_T: uint32_t
AudioSignalActivation_T: uint8_t
AudioTerminationCause_T: uint8_t
AudioVolume_T: uint8_t
ButtonOptions_T: uint8_t
ChangeActiveMask_ErrorCode_T: uint8_t
ChangeAttribute_ErrorCode_T: uint8_t
ChangeBackgroundColour_ErrorCode_T: uint8_t
ChangeChildLocation_ErrorCode_T: uint8_t
ChangeChildPosition_ErrorCode_T: uint8_t
ChangeEndPoint_ErrorCode_T: uint8_t
ChangeFillAttributes_ErrorCode_T: uint8_t
ChangeFontAttributes_ErrorCode_T: uint8_t
ChangeLineAttributes_ErrorCode_T: uint8_t
ChangeListItem_ErrorCode_T: uint8_t
ChangeNumericValue_ErrorCode_T uint8_t
ChangeObjectLabel_ErrorCode_T: uint8_t
ChangePolygonPoint_ErrorCode_T: uint8_t
ChangePolygonScale_ErrorCode_T: uint8_t
ChangePriority_ErrorCode_T: uint8_t
ChangeSize_ErrorCode_T: uint8_t
ChangeSoftKeyMask_ErrorCode_T: uint8_t
ChangeStringValue_ErrorCode_T: uint8_t
ControlAudioSignal_ErrorCode_T: uint8_t
DeleteVersion_ErrorCode_T: uint8_t
EnableDisableObject_ErrorCode_T: uint8_t
EllipseType_T: uint8_t
Esc_ErrorCode_T: uint8_t
ExecuteMacro_ErrorCode_T: uint8_t
FontSize_T: uint8_t
FontStyle_T: uint8_t
FontType_T: uint8_t

GetAttributeValue_ErrorCode_T: uint8_t
GetMemory_Status_T: uint8_t
GetSupportedWideChars_ErrorCode_T: uint8_t
GraphicTicks_T: uint8_t
GraphicsContext_ErrorCode_T: uint8_t
GraphicsContextOptions_T: uint8_t
GraphicsZoom_T: float32_t
HideShowObject_ErrorCode_T: uint8_t
Justification_T: uint8_t
KeyCode_T: uint8_t
KeyGroupOptions_T: uint8_t
LinearBarGraphOptions_T: uint8_t
LineArt_T: uint16_t
LineDirection_T: uint8_t
LineSuppression_T: uint8_t
ListIndex_T: uint8_t
LockUnlockMask_ErrorCode_T: uint8_t
MeterOptions_T: uint8_t
ModelIdentificationCode_T: uint16_t
NumberFormat_T: uint8_t
NumberOfDecimals_T: uint8_t
NumberOffset_T: int32_t
NumberOptions_T: uint8_t
NumberScaleFactor_T: float32_t
NumericValue_T: uint32_t
PictureGraphicOptions_T: uint8_t
PolygonPointIndex_T: uint8_t
PolygonType_T: uint8_t
SelectColourMap_ErrorCode_T: uint8_t
SelectInputObject_ErrorCode_T: uint8_t
SetAudioVolume_ErrorCode_T: uint8_t
StatusBusyCodes_T: uint8_t
StoreVersion_ErrorCode_T: uint8_t
StringOptions_T: uint8_t
SupportedFonts_T: uint16_t
VT_ChangeActiveMask_ErrorCode_T: uint8_t
VT_ChangeSoftKeyMask_ErrorCode_T: uint8_t
VT_ESC_ErrorCode_T: uint8_t
VT_Features_T: uint8_t
VT_Version_T: uint8_t
WideChar_CodePlane_T: uint8_t
WideChar_T: uint16_t
WindowMaskOptions_T: uint8_t

Enumerations

EnableDisable_Status_T

Enumeration for Enable/Disable Object response

Signature

```
typedef enum EnableDisable_Status_E EnableDisable_Status_T
```

Members

Object_Disabled
Object is disabled

Object_Enabled
Object is enabled

FillType_T

Enumeration for Fill Attributes Fill Type

Signature
```
typedef enum FillType_E FillType_T
```

Members
`FILL_NO_FILL` : No fill
`FILL_LINE_COLOUR` : Fill with line colour
`FILL_ATTRIBUTE` : Fill with specified colour in fill colour attribute
`FILL_PATTERN` : Fill with pattern given by fill pattern attribute

KeyButton_ActivationCode_T

Enumeration for activation code for buttons/softkeys

Signature
```
typedef enum KeyButton_ActivationCode_E KeyButton_ActivationCode_T
```

Members
`KeyButton_Released` : Softkey or Button Released (State change)
`KeyButton_Pressed` : Softkey or Button Pressed (State change)
`KeyButton_Held` : Softkey or Button Held (Not state change)
`KeyButton_Aborted` : Softkey or Button press aborted (finger moved off button without releasing)

MaskCommand_T

Enumeration for locking/unlocking Data Mask or Window Mask

Signature
```
typedef enum MaskCommand_E MaskCommand_T
```

Members
`Mask_Unlock` : Unlock Data Mask or Window Mask
`Mask_Lock` : Lock Data Mask or Window Mask

`MaskType_T`

Enumeration for type of Mask (Data or Alarm)

Signature
```
typedef enum MaskType_E MaskType_T
```

Members
`Mask_DataMask` : Data Mask
`Mask_AlarmMask` : Alarm Mask

`Object_SelectionState_T`

Enumeration for Selection State of Input Object

Signature
```
typedef enum Object_SelectionState_E Object_SelectionState_T
```

Members
`Object_NotSelected` : Object is not selected
`Object_Selected` : Object is selected (but not opened for input)
`Object_SelectedAndOpenForEdit` : Object is selected and opened for input

`ObjectPool_ScaleFactor_T`

Enumeration to indicate how to scale an object pool part (by data mask size or soft key size)

Signature
```
typedef enum ObjectPool_ScaleFactor_E ObjectPool_ScaleFactor_T
```

Members
`ScaleFactor_None` : Do not scale this object pool part
`ScaleFactor_DataMask` : Scale based on the Data Mask size
`ScaleFactor_SoftKeyMask` : Scale based on the Soft Key size

`PointingEvent_TouchState_T`

Enumeration for Pointing Event Touch State

Signature
```
typedef enum PointingEvent_TouchState_E PointingEvent_TouchState_T
```

Members
`TouchState_Released` : Screen Location Released (State change)
`TouchState_Pressed` : Screen Location Pressed (State change)
`TouchState_Held` : Screen Location Held (Not state change)

ShowHide_Status_T

Enumeration for Hide/Show Object response

Signature
```
typedef enum ShowHide_Status_E ShowHide_Status_T
```

Members
`Object_Hidden` : Object is not visible
`Object_Shown` : Object is visible

VT_GraphicType_T

Enumeration values for supported graphic modes (colour depth)

Signature
```
typedef enum VT_GraphicType_E VT_GraphicType_T
```

Members
`VT_Monochrome` : VT supports colour codes 0 and 1 and monochrome Picture Graphic objects only
`VT_16_Colour` : VT supportes colour codes 0 to 15 and monochrome and 16 colour Picture Graphic objects
`VT_256_Colour` : VT supportes colour codes 0 to 255 and monochrome and all formats of Picture Graphic objects

Structures

ButtonActivation_T

Structure to hold Button Activation message data.

Signature
```
typedef struct ButtonActivation_S ButtonActivation_T
```

Members
`ObjectID_T key_object_id` : Object ID of Button Object
`ObjectID_T parent_object_id` : Object ID of parent Data Mask or in the case where the Button is in a visible Window Mask object, the Object ID of the Window Mask object
`KeyButton_ActivationCode_T  key_activation_code` : Key activation code
`KeyCode_T  button_key_code` : Button key code

ButtonActivation_Callback_T

Structure for registering Button Activation callback.

Signature
```
typedef struct ButtonActivation_Callback_S ButtonActivation_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ButtonActivation)(VTClient_T *vt_client, const VT_T *vt, const`

ButtonActivation_T *) : Callback function pointer
struct LinkedList_Node_S Node : Linked List node

ChangeActiveMask_Response_T

Structure to hold Change Active Mask response data.

Signature
```
typedef struct ChangeActiveMask_Response_S ChangeActiveMask_Response_T
```

Members
ObjectID_T object_id : Object ID
ChangeActiveMask_ErrorCode_T error_code : Error Codes (0 = no errors)

ChangeActiveMask_Response_Callback_T

Structure for registering Change Active Mask callback.

Signature
```
typedef struct ChangeActiveMask_Response_Callback_S
ChangeActiveMask_Response_Callback_T
```

Members
const ObjectID_T object_id : Object ID
const void (*ChangeActiveMask_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeActiveMask_Response_T *) : Callback function pointer
struct LinkedList_Node_S Node : Linked List node

ChangeAttribute_Response_Callback_T

Structure for registering Change Attribute callback.

Signature
```
typedef struct ChangeAttribute_Response_Callback_S
ChangeAttribute_Response_Callback_T
```

Members
const ObjectID_T object_id : Object ID
const AttributeID_T AttributeID : Attribute ID const void (*ChangeAttribute_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeAttribute_Response_T *) : Callback function pointer
struct LinkedList_Node_S Node : Linked List node

ChangeBackgroundColour_Response_T

Structure to hold Change Background Colour response data.

Signature
```
typedef struct ChangeBackgroundColour_Response_S
ChangeBackgroundColour_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`Colour_T new_color` : New Background colour
`ChangeBackgroundColour_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeBackgroundColour_Response_Callback_T

Structure for registering Change Background Colour callback.

Signature
```
typedef struct ChangeBackgroundColour_Response_Callback_S
ChangeBackgroundColour_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeBackgroundColour_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeBackgroundColour_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeChildLocation_Response_T

Structure to hold Change Child Location response data.

Signature
```
typedef struct ChangeChildLocation_Response_S ChangeChildLocation_Response_T
```

Members
`ObjectID_T parent_object_id` : Parent Object ID
`ObjectID_T object_id` : Object ID of object to move
`ChangeChildLocation_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeChildLocation_Response_Callback_T

Structure for registering Child Location callback.

Signature
```
typedef struct ChangeChildLocation_Response_Callback_S
ChangeChildLocation_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeChildLocation_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeChildLocation_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeChildPosition_Response_T

Structure to hold Change Child Position response data.

Signature
```
typedef struct ChangeChildPosition_Response_S ChangeChildPosition_Response_T
```

Members
`ObjectID_T parent_object_id` : Parent Object ID
`ObjectID_T object_id` : Object ID of object to move
`ChangeChildPosition_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangeChildPosition_Response_Callback_T`

Structure for registering Change Child Position callback.

Signature
```
typedef struct ChangeChildPosition_Response_Callback_S
ChangeChildPosition_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeChildPosition_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeChildPosition_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangeEndPoint_Response_T`

Structure to hold Change End Point response data.

Signature
```
typedef struct ChangeEndPoint_Response_S ChangeEndPoint_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`ChangeEndPoint_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangeEndPoint_Response_Callback_T`

Structure for registering Change End Point callback.

Signature
```
typedef struct ChangeEndPoint_Response_Callback_S
ChangeEndPoint_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeEndPoint_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeEndPoint_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangeFillAttributes_Response_T`

Structure to hold Change Fill Attributes response data.

Signature
```
typedef struct ChangeFillAttributes_Response_S
ChangeFillAttributes_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`ChangeFillAttributes_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeFillAttributes_Response_Callback_T

Structure for registering Change Fill Attributes callback.

Signature
```
typedef struct ChangeFillAttributes_Response_Callback_S
ChangeFillAttributes_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeFillAttributes_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeFillAttributes_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeFontAttributes_Response_T

Structure to hold Change Font Attributes response data.

Signature
```
typedef struct ChangeFontAttributes_Response_S
ChangeFontAttributes_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`ChangeFontAttributes_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeFontAttributes_Response_Callback_T

Structure for registering Change Font Attributes callback.

Signature
```
typedef struct ChangeFontAttributes_Response_Callback_S
ChangeFontAttributes_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeFontAttributes_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeFontAttributes_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeLineAttributes_Response_T

Structure to hold Change Line Attributes response data.

Signature
```
typedef struct ChangeLineAttributes_Response_S
ChangeLineAttributes_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`ChangeLineAttributes_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangeLineAttributes_Response_Callback_T`

Structure for registering Change Line Attributes callback.

Signature
```
typedef struct ChangeLineAttributes_Response_Callback_S
ChangeLineAttributes_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeLineAttributes_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeLineAttributes_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangeListItem_Response_T`

Structure to hold Change List Item response data.

Signature
```
typedef struct ChangeListItem_Response_S ChangeListItem_Response_T
```

Members
`ObjectID_T object_id` : Object ID of an Input List object or Output List object
`ListIndex_T list_index` : List Index (items are numbered 0-n)
`ObjectID_T new_object_id` : New Object ID or NULL_OBJECT_ID to set empty item
`ChangeListItem_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangeListItem_Response_Callback_T`

Structure for registering Change List Item callback.

Signature
```
typedef struct ChangeListItem_Response_Callback_S
ChangeListItem_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeListItem_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeListItem_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangeNumericValue_Response_T`

Structure to hold Change Numeric Value response data.

Signature
```
typedef struct ChangeNumericValue_Response_S ChangeNumericValue_Response_T
```

Members

ObjectID_T object_id
Object ID

NumericValue_T value
Value: Size depends on object type. Objects of size 1 byte are found in byte 5. Objects of size 2 bytes are found in Bytes 5-6. Values greater than 1 byte are transmitted little endian (LSB first)

ChangeNumericValue_ErrorCode_T error_code
Error Codes (0 = no errors)

ChangeNumericValue_Response_Callback_T

Structure for registering Change Numeric Value callback.

Signature

```
typedef struct ChangeNumericValue_Response_Callback_S
ChangeNumericValue_Response_Callback_T
```

Members

const ObjectID_T object_id
Object ID

const void (*ChangeNumericValue_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeNumericValue_Response_T *)
Callback function pointer

struct LinkedList_Node_S Node
Linked List node

ChangeObjectLabel_Response_T

Structure to hold Change Object Label response data.

Signature
```
typedef struct ChangeObjectLabel_Response_S ChangeObjectLabel_Response_T
```

Members
`ChangeObjectLabel_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangePolygonPoint_Response_T

Structure to hold Change Polygon Point response data.

Signature
```
typedef struct ChangePolygonPoint_Response_S ChangePolygonPoint_Response_T
```

Members
`ObjectID_T object_id` : Object ID of the Polygon object to change
`ChangePolygonPoint_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangePolygonPoint_Response_Callback_T`

Structure for registering Change Polygon Point callback.

Signature
```
typedef struct ChangePolygonPoint_Response_Callback_S
ChangePolygonPoint_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangePolygonPoint_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangePolygonPoint_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangePolygonScale_Response_T`

Structure to hold Change Polygon Scale response data.

Signature
```
typedef struct ChangePolygonScale_Response_S ChangePolygonScale_Response_T
```

Members
`ObjectID_T mask_object_id` : Object ID of Polygon object
`Pixel_T new_width_attribute` : New width attribute
`Pixel_T new_heigh_attribute` : New height attribute
`ChangePolygonScale_ErrorCode_T error_code` : Error Codes (0 = no errors)

`ChangePolygonScale_Response_Callback_T`

Structure for registering Change Polygon Scale callback.

Signature
```
typedef struct ChangePolygonScale_Response_Callback_S
ChangePolygonScale_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangePolygonScale_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangePolygonScale_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`ChangePriority_Response_Callback_T`

Structure for registering Change Priority callback.

Signature
```
typedef struct ChangePriority_Response_Callback_S
ChangePriority_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangePriority_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangePriority_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeSize_Response_T

Structure to hold Change Size response data.

Signature
`typedef struct ChangeSize_Response_S ChangeSize_Response_T`

Members
`ObjectID_T object_id` : Object ID of the object to change
`ChangeStringValue_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeSize_Response_Callback_T

Structure for registering Change Size callback.

Signature
`typedef struct ChangeSize_Response_Callback_S ChangeSize_Response_Callback_T`

Members
`const ObjectID_T object_id` : Object ID
`const void (*ChangeSize_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeSize_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeSoftKeyMask_Response_T

Structure to hold Change Soft Key Mask response data.

Signature
`typedef struct ChangeSoftKeyMask_Response_S ChangeSoftKeyMask_Response_T`

Members
`ObjectID_T mask_object_id` : Data or Alarm Mask Object ID
`ObjectID_T soft_key_mask_object_id` : Soft Key Mask Object ID
`ChangeSoftKeyMask_ErrorCode_T error_code` : Error Codes (0 = no errors)

ChangeSoftKeyMask_Response_Callback_T

Structure for registering Change Soft Key Mask callback.

Signature
`typedef struct ChangeSoftKeyMask_Response_Callback_S ChangeSoftKeyMask_Response_Callback_T`

Members
`const ObjectID_T object_id` : Object ID

`const void (*ChangeSoftKeyMask_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeSoftKeyMask_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ChangeStringValue_Response_T

Structure to hold Change String Value response data.

Signature

```
typedef struct ChangeStringValue_Response_S ChangeStringValue_Response_T
```

Members

`ObjectID_T object_id`
Object ID

`NumericValue_T value`
Value: Size depends on object type. Objects of size 1 byte are found in byte 5. Objects of size 2 bytes are found in Bytes 5-6. Values greater than 1 byte are transmitted little endian (LSB first)

`ChangeNumericValue_ErrorCode_T error_code`
Error Codes (0 = no errors)

ChangeStringValue_Response_Callback_T

Structure for registering Change String Value callback.

Signature

```
typedef struct ChangeStringValue_Response_Callback_S
ChangeStringValue_Response_Callback_T
```

Members

`const ObjectID_T object_id` : Object ID
`const void (*ChangeStringValue_Response)(VTClient_T *vt_client, const VT_T *vt, const ChangeStringValue_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ControlAudioSignal_Response_T

Structure to hold Control Audio Signal response data.

Signature

```
typedef struct ControlAudioSignal_Response_S ControlAudioSignal_Response_T
```

Members

`ControlAudioSignal_ErrorCode_T error_code` : Error Codes (0 = no errors)

DeleteVersion_Response_T

Structure to hold Delete Version response.

Signature
`typedef struct DeleteVersion_Response_S DeleteVersion_Response_T`

Members
`DeleteVersion_ErrorCode_T error_code` : Error Codes (0 = no errors)

`EnableDisableObject_Response_T`

Structure to hold Enable/Disable Object response data.

Signature
`typedef struct EnableDisableObject_Response_S EnableDisableObject_Response_T`

Members
`ObjectID_T object_id` : Object ID
`EnableDisable_Status_T enable_disable_status` : Object Enabled state
`EnableDisableObject_ErrorCode_T error_code` : Error Codes (0 = no errors)

`EnableDisableObject_Response_Callback_T`

Structure for registering Enable/Disable Object callback.

Signature
`typedef struct EnableDisableObject_Response_Callback_S EnableDisableObject_Response_Callback_T`

Members
`const ObjectID_T object_id` : Object ID
`const void (*EnableDisableObject_Response)(VTClient_T *vt_client, const VT_T *vt, const EnableDisableObject_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`Esc_Response_T`

Structure to hold ESC response data.

Signature
`typedef struct Esc_Response_S Esc_Response_T`

Members
`ObjectID_T object_id` : Object ID where input was aborted if no error code
`Esc_ErrorCode_T error_code` : Error Codes (0 = no errors)

`Esc_Response_Callback_T`

Structure for registering ESC callback.

Signature
`typedef struct Esc_Response_Callback_S Esc_Response_Callback_T`

Members
`const ObjectID_T object_id` : Object ID

`const void (*Esc_Response)(VTClient_T *vt_client, const VT_T *vt, const Esc_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

ExecuteMacro_Response_T

Structure to hold Execute Macro response data.

Signature
```
typedef struct ExecuteMacro_Response_S ExecuteMacro_Response_T
```

Members
`MacroID_T macro_id` : Object ID of Macro object
`ExecuteMacro_ErrorCode_T error_code` : Error Codes (0 = no errors)

GetAttributeValue_Response_T

Structure to hold Get Attribute Value response data.

Signature
```
typedef struct GetAttributeValue_Response_S GetAttributeValue_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`AttributeID_T attribute_id` : Attribute ID of the Object
`GetAttributeValue_ErrorCode_T error_code` : Error Codes (0 = no errors)
`AttributeValue_T value` : Current value of the attribute. Size depends on attribute data type. Values greater than 1 byte are transmitted little endian (LSB first):
Boolean: 1 byte for TRUE/FALSE
Integer: 1, 2 or 4 bytes as defined in object tables
Float: 4 bytes
Bitmask: 1 byte

GetAttributeValue_Response_Callback_T

Structure for registering Get Attribute Value callback.

Signature
```
typedef struct GetAttributeValue_Response_Callback_S
GetAttributeValue_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const AttributeID_T AttributeID` : Attribute ID
`const void (*GetAttributeValue_Response)(VTClient_T *vt_client, const VT_T *vt, const GetAttributeValue_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

GetHardware_Response_T

Structure to hold Get Hardware response data.

Signature
`typedef struct GetHardware_Response_S GetHardware_Response_T`

Members
`Time_T BootTime` : Max time from power cycle to transmission of first "VT Status message" (0xFF when not available)
`VT_GraphicType_T GraphicType` : Supported graphic modes
`VT_Features_T Features` : Supported Hardware Features
`Pixel_T DataMask_X_Pixels` : Number of X-axis pixels in data mask
`Pixel_T DataMask_Y_Pixels` : Number of Y-axis pixels in data mask

GetMemory_Response_T

Structure to hold Get Memory Response message data.

Signature
`typedef struct GetMemory_Response_S GetMemory_Response_T`

Members
`VT_Version_T Version` : VT Version
`GetMemory_Status_T Status` : Get Memory Status

GetSoftKeys_Response_T

Structure to hold Get Number of Soft Keys response data.

Signature
`typedef struct GetSoftKeys_Response_S GetSoftKeys_Response_T`

Members
`Pixel_T X_Pixels` : Number of pixels on the X-axis for a Soft Key descriptor
`Pixel_T Y_Pixels` : Number of pixels on the Y-axis for a Soft Key descriptor
`SoftKeyCount_T Physical` : Number of Physical Soft Keys
`SoftKeyCount_T Virtual` : Number of possible virtual Soft Keys in a Soft Key Mask
`SoftKeyCount_T Navigation` : Number of Physical Soft Keys used by the VT for navigation among Virtual Soft Keys (VTv4 and later)

GetSupportedObjects_Response_T

Structure to hold Get Supported Objects response data.

Signature
`typedef struct GetSupportedObjects_Response_S GetSupportedObjects_Response_T`

Members
`Size_T NumberOfBytes` : Number of bytes in SupportedObjects
`Pipe_ReadHandle_T SupportedObjects` : Numerically ascending sorted list of all Object Types supported by the VT

GetSupportedWideChars_Response_T

Structure to hold Get Supported WideChars response data.

Signature
```
typedef struct GetSupportedWideChars_Response_S
GetSupportedWideChars_Response_T
```

Members
`WideChar_CodePlane_T CodePlane` : WideChar Code Plane
`WideChar_T FirstWideChar` : First WideChar in inquiry range
`WideChar_T LastWideChar` : Last WideChar in inquiry range
`GetSupportedWideChars_ErrorCode_T ErrorCode` : Error Codes (0 = no errors)
`Size_T NumberOfRanges` : Indicates the number of entries in the WideChar range array. Set to zero if Error codes is not equal to 0
`Pipe_ReadHandle_T WideCharRanges` : Each entry in the array consists of two WideChars: first WideChar, last WideChar

GetTextFont_Response_T

Structure to hold Get Text Font Data response data.

Signature
```
typedef struct GetTextFont_Response_S GetTextFont_Response_T
```

Members
`SupportedFonts_T Sizes` : Supported font sizes
`FontStyle_T Styles` : Supported font styles

GetVersions_Response_T

Structure to hold Get Version response.

Signature
```
typedef struct GetVersions_Response_S GetVersions_Response_T
```

Members
`Size_T NumberOfVersions` : Number of Version Labels in the response (7 bytes/characters each)
`Pipe_ReadHandle_T Versions` : Pipe handle to the list of received Version Labels

GetWindowMask_Response_T

Structure to hold Get Window Mask Data response data.

Signature
```
typedef struct GetWindowMask_Response_S GetWindowMask_Response_T
```

Members
`Colour_T UserLayoutBackgroundColour` : Background colour of VT's User-Layout Data Masks

`Colour_T KeyCellBackgroundColour` : Background colour of VT's Key Cells when on a User-Layout Soft Key Mask

`GraphicsContext_Response_T`

Structure to hold Graphics Context response data.

Signature
```
typedef struct GraphicsContext_Response_S GraphicsContext_Response_T
```

Members
`ObjectID_T object_id` : Object ID of a Graphics Context object
`GraphicsContext_Subcommand_T sub_command_id` : Sub-command ID
`GraphicsContext_ErrorCode_T error_code` : Error Codes (0 = no errors)

`GraphicsContext_Response_Callback_T`

Structure for registering Graphics Context callback.

Signature
```
typedef struct GraphicsContext_Response_Callback_S
GraphicsContext_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const GraphicsContext_Subcommand_T SubcommandID` : Subcommand ID
`const void (*GraphicsContext_Response)(VTClient_T *vt_client, const VT_T *vt, const GraphicsContext_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`HideShowObject_Response_T`

Structure to hold Hide/Show Object response data.

Signature
```
typedef struct HideShowObject_Response_S HideShowObject_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`ShowHide_Status_T hide_show_status` : Object visibility
`HideShowObject_ErrorCode_T error_code` : Error Codes (0 = no errors)

`HideShowObject_Response_Callback_T`

Structure for registering Hide/Show Object callback.

Signature
```
typedef struct HideShowObject_Response_Callback_S
HideShowObject_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID

`const void (*HideShowObject_Response)(VTClient_T *vt_client, const VT_T *vt, const HideShowObject_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

LoadVersion_Response_T

Structure to hold Load Version response.

Signature
```
typedef struct LoadVersion_Response_S LoadVersion_Response_T
```

Members
`LoadVersion_ErrorCode_T error_code` : Error Codes (0 = no errors)

LockUnlockMask_Response_T

Structure to hold Lock/Unlock Mask response data.

Signature
```
typedef struct LockUnlockMask_Response_S LockUnlockMask_Response_T
```

Members
`MaskCommand_T command` : Command
`LockUnlockMask_ErrorCode_T error_code` : Error Codes (0 = no errors)

ObjectPool_T

Structure to hold combined Object Pool Parts data in a complete Object Pool.

Signature
```
typedef struct ObjectPool_S ObjectPool_T
```

Members
`ObjectPoolPart_T  const *Parts` : Data for each part of the object pool
`Size_T NumParts` : Number of parts to the object pool
`Pixel_T DataMask_XY` : Designed data mask size
`Pixel_T SoftKey_X` : Designed softkey mask width
`Pixel_T SoftKey_Y` : Designed softkey mask height
`char VersionLabel[8]` : Application's object pool version string
`char DefaultLanguage[3]` : Default language for this object pool

ObjectPoolPart_T

Structure to hold Object Pool information data.

Signature
```
typedef struct ObjectPoolPart_S ObjectPoolPart_T
```

Members
`MemoryPointer_T Data` : Object Pool Data
`Size_T Size` : Size of Object Pool Data

`ObjectPool_ScaleFactor_T ScaleFactor` : How to scale this part
`char Language[3]` : Language specific part (" " if not specific)

PointingEvent_T

Structure to hold Pointing Event message data.

Signature
```
typedef struct PointingEvent_S PointingEvent_T
```

Members
`Pixel_T x_position` : X Position in pixels relative to top left corner of Data Mask area
`Pixel_T y_position` : Y Position in pixels relative to top left corner of Data Mask area
`PointingEvent_TouchState_T touch_state` : Touch State

SelectColourMap_Response_T

Structure to hold Select Colour Map response data.

Signature
```
typedef struct SelectColourMap_Response_S SelectColourMap_Response_T
```

Members
`ObjectID_T object_id` : Object ID of the Colour Map object
`SelectColourMap_ErrorCode_T error_code` : Error Codes (0 = no errors)

SelectColourMap_Response_Callback_T

Structure for registering Select Colour Map callback.

Signature
```
typedef struct SelectColourMap_Response_Callback_S
SelectColourMap_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*SelectColourMap_Response)(VTClient_T *vt_client, const VT_T *vt, const SelectColourMap_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

SelectInputObject_Response_T

Structure to hold Select Input Object response data.

Signature
```
typedef struct SelectInputObject_Response_S SelectInputObject_Response_T
```

Members
`ObjectID_T object_id` : Object ID
`Object_SelectionState_T SelectState` : Input Selection State
`SelectInputObject_ErrorCode_T error_code` : Error Codes (0 = no errors)

SelectInputObject_Response_Callback_T

Structure for registering Select Input Object callback.

Signature
```
typedef struct SelectInputObject_Response_Callback_S
SelectInputObject_Response_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*SelectInputObject_Response)(VTClient_T *vt_client, const VT_T *vt, const SelectInputObject_Response_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

SetAudioVolume_Response_T

Structure to hold Set Audio Volume response data.

Signature
```
typedef struct SetAudioVolume_Response_S SetAudioVolume_Response_T
```

Members
`SetAudioVolume_ErrorCode_T error_code` : Error Codes (0 = no errors)

SoftKeyActivation_T

Structure to hold Soft Key Activation message data.

Signature
```
typedef struct SoftKeyActivation_S SoftKeyActivation_T
```

Members
`ObjectID_T key_object_id` : Object ID of Key Object
`ObjectID_T parent_object_id` : Object ID of visible Data Mask, Alarm Mask, or in the case where the Soft Key is in a visible Key Group, the Object ID of the Key Group Object
`KeyButton_ActivationCode_T key_activation_code` : Key activation code
`KeyCode_T soft_key_code` : Soft key code

SoftKeyActivation_Callback_T

Structure for registering Soft Key Activation callback.

Signature
```
typedef struct SoftKeyActivation_Callback_S SoftKeyActivation_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*SoftKeyActivation)(VTClient_T *vt_client, const VT_T *vt, const SoftKeyActivation_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

StoreVersion_Response_T

Structure to hold Store Version response.

Signature
```
typedef struct StoreVersion_Response_S StoreVersion_Response_T
```

Members
`StoreVersion_ErrorCode_T error_code` : Error Codes (0 = no errors)

VT_T

Structure to hold known data for a VT.

Signature

```
typedef struct VT_S VT_T
```

Members

Mutex_T Mutex
Mutex for VT data

VT_Status_T Status
Data for the VT Status messages (ISO 11783-6 Annex G)

VT_Connection_T Connection
Data for VT connection management

VT_Metrics_T Metrics
Data for the various Technical Data messages (ISO 11783-6 Annex D)

VT_ObjectPool_T ObjectPool
Loaded Object Pool

VtChangeActiveMask_Callback_T

Structure for registering VT Change Active Mask callback.

Signature
```
typedef struct VtChangeActiveMask_Callback_S VtChangeActiveMask_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtChangeActiveMask)(VTClient_T *vt_client, const VT_T *vt, const VtChangeActiveMask_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

VtChangeNumericValue_Callback_T

Structure for registering VT Select Input Object callback.

Signature

```
typedef struct VtChangeNumericValue_Callback_S
VtChangeNumericValue_Callback_T
```

Members

`const ObjectID_T object_id`
Object ID

`const void (*VtChangeNumericValue)(VTClient_T *vt_client, const VT_T *vt, const VtChangeNumericValue_T *)`
Callback function pointer

`struct LinkedList_Node_S Node`
Linked List node

`VtChangeSoftKeyMask_Callback_T`

Structure for registering VT Change Soft Key Mask callback.

Signature
```
typedef struct VtChangeSoftKeyMask_Callback_S VtChangeSoftKeyMask_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtChangeSoftKeyMask)(VTClient_T *vt_client, const VT_T *vt, const VtChangeSoftKeyMask_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`VtChangeStringValue_Callback_T`

Structure for registering VT Change Soft Key Mask callback.

Signature
```
typedef struct VtChangeStringValue_Callback_S VtChangeStringValue_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtChangeStringValue)(VTClient_T *vt_client, const VT_T *vt, VtChangeStringValue_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

`VtEsc_Callback_T`

Structure for registering VT ESC callback.

Signature
```
typedef struct VtEsc_Callback_S VtEsc_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtEsc)(VTClient_T *vt_client, const VT_T *vt, const VtEsc_T *)` :

Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

VtOnUserLayoutHideShow_Callback_T

Structure for registering VT Change Soft Key Mask callback.

Signature
```
typedef struct VtOnUserLayoutHideShow_Callback_S
VtOnUserLayoutHideShow_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtOnUserLayoutHideShow)(VTClient_T *vt_client, const VT_T *vt, const VtOnUserLayoutHideShow_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

VtSelectInputObject_Callback_T

Structure for registering VT Select Input Object callback.

Signature
```
typedef struct VtSelectInputObject_Callback_S VtSelectInputObject_Callback_T
```

Members
`const ObjectID_T object_id` : Object ID
`const void (*VtSelectInputObject)(VTClient_T *vt_client, const VT_T *vt, const VtSelectInputObject_T *)` : Callback function pointer
`struct LinkedList_Node_S Node` : Linked List node

VTv4_T

Structure to hold supported objects/commands for VTv4.

Signature
```
typedef struct VTv4_S VTv4_T
```

Members

`VTv4_WorkingSet_T WorkingSet`
Working Set object commands

`VTv4_DataMask_T DataMask`
Data Mask object commands

`VTv4_AlarmMask_T AlarmMask`
Alarm Mask object commands

`VTv4_Container_T Container`
Container object commands

VTv4_SoftKeyMask_T SoftKeyMask
Soft Key Mask object commands

VTv4_Key_T Key
Key object commands

VTv4_Button_T Button
Button object commands

VTv4_InputBoolean_T InputBoolean
Input Boolean object commands

VTv4_InputString_T InputString
Input String object commands

VTv4_InputNumber_T InputNumber
Input Number object commands

VTv4_InputList_T InputList
Input List object commands

VTv4_OutputString_T OutputString
Output String object commands

VTv4_OutputNumber_T OutputNumber
Output Number object commands

VTv4_OutputList_T OutputList
Output List object commands

VTv4_Line_T Line
Line object commands

VTv4_Rectangle_T Rectangle
Rectangle object commands

VTv4_Ellipse_T Ellipse
Ellipse object commands

VTv4_Polygon_T Polygon
Polygon object commands

VTv4_Meter_T Meter
Meter object commands

VTv4_LinearBarGraph_T LinearBarGraph
Linear Bar Graph object commands

VTv4_ArchedBarGraph_T ArchedBarGraph
Arched Bar Graph object commands

VTv4_PictureGraphic_T PictureGraphic
Picture Graphic object commands

VTv4_NumberVariable_T NumberVariable
Number Variable object commands

VTv4_StringVariable_T StringVariable
String Variable object commands

VTv4_FontAttributes_T FontAttributes
Font Attributes object commands

VTv4_LineAttributes_T LineAttributes
Line Attributes object commands

VTv4_FillAttributes_T FillAttributes
Fill Attributes object commands

VTv4_InputAttributes_T InputAttributes
Input Attributes object commands

VTv4_ExtendedInputAttributes_T ExtendedInputAttributes
Extended Input Attributes object commands

VTv4_ObjectPointer_T ObjectPointer
Object Pointer object commands

VTv4_Macro_T Macro
Macro object commands

VTv4_ColourMap_T ColourMap
Colour Map object commands

VTv4_GraphicsContext_T GraphicsContext
Graphics Context object commands

VTv4_WindowMask_T WindowMask
Window Mask object commands

VTv4_KeyGroup_T KeyGroup
Key Group object commands

VTv4_ObjectLabelReferenceList_T ObjectLabelReferenceList
Object Label Reference List object commands

VTv4_AlarmMask_T

Structure to hold supported commands for a VTv4 Alarm Mask object.

Signature

```
typedef struct VTv4_AlarmMask_S VTv4_AlarmMask_T
```

Members

bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, Colour_T background_colour)
Change Background Colour command

bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position)
Change Child Location command (Relative Position)

bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T data_mask, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position)
Change Child Position command (Absolute Position)

bool_t (*ChangePriority)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, AlarmPriority_T priority)
Change Priority command

bool_t (*ChangeSoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, ObjectID_T soft_key_mask)
Change Soft Key Mask command

struct ChangeAttribute
Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, Colour_T background_colour)

Change Background Colour attribute

bool_t (*SoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T soft_key_mask)
Change Soft Key Mask attribute

bool_t (*Priority)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, AlarmPriority_T priority)
Change Priority attribute

bool_t (*AcousticSignal)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, AcousticSignal_T priority)
Change Acoustic Signal attribute
}

struct GetAttributeValue
Get Attribute Command
{
`bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set)`

Get Type attribute

`bool_t (*BackgroundColour) (const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set)`
Get Background Colour attribute

`bool_t (*SoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask)`
Get Soft Key Mask attribute

`bool_t (*Priority)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, AlarmPriority_T priority)`
Change Priority attribute

`bool_t (*AcousticSignal)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, AcousticSignal_T priority)`
Change Acoustic Signal attribute
}

VTv4_ArchedBarGraph_T

Structure to hold supported commands for a VTv4 Arched Bar Graph object.

Signature
`typedef struct VTv4_ArchedBarGraph_S VTv4_ArchedBarGraph_T`

Members
`bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, NumericValue_T value)` : Change Numeric Value attribute
`bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type)` : Change Size attribute
`struct ChangeAttribute` : Change Attribute command
{
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width)` : Change Width attribute
`bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Pixel_T height)` : Change Height attribute
`bool_t (*Colour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Colour_T colour)` :

Change Colour attribute
bool_t (*TargetLineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Colour_T target_line_colour) : Change Target Line Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, LinearBarGraphOptions_T options) : Change Options attribute
bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph, Angle_T start) : Change Start Angle attribute
bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph, Angle_T end) : Change End Angle attribute
bool_t (*BarGraphWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph, Pixel_T width) : Change Bar Graph Width attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph, NumericValue_T minimum_value) : Change Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, NumericValue_T maximum_value) : Change Maximum Value attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, ObjectID_T variable) : Change Variable Reference attribute
bool_t (*TargetValueReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, ObjectID_T target_variable) : Change Target Value Reference attribute
bool_t (*TargetValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, NumericValue_T target_value) : Change Target Value attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Height attribute
bool_t (*Colour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Colour attribute
bool_t (*TargetLineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Target Line Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Options attribute

bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Start Angle attribute
bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get End Angle attribute
bool_t (*BarGraphWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Bar Graph Width attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Maximum Value attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Variable Reference attribute
bool_t (*TargetValueReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Target Value Reference attribute
bool_t (*TargetValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Target Value attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T arched_bar_graph) : Get Value attribute
}

VTv4_Button_T

Structure to hold supported commands for a VTv4 Button object.

Signature
```
typedef struct VTv4_Button_S VTv4_Button_T
```

Members
bool_t (*EnableDisableObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, EnableDisable_Status_T enable_flag) : Enable/Disable Object command VTv4 and later only
bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Object_SelectionState_T option) : Select Input Object command VTv4 and later only
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Pixel_T width, Pixel_T height) : Change Size command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt,

`const ISOBUS_Callback_T *callback, ObjectID_T button, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position)` : Change Child Location command (Relative Position)
`bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T button, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position)` : Change Child Position command (Absolute Position)
`struct ChangeAttribute` : Change Attribute command
{
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Pixel_T width)` : Change Width attribute
`bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Pixel_T height);` : Change Height attribute
`bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Colour_T background_colour)` : Change Background Colour attribute
`bool_t (*BorderColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, Colour_T border_colour)` : Change Border Colour attribute
`bool_t (*KeyCode)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, KeyCode_T key_code)` : Change Key Code attribute
`bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button, ButtonOptions_T options)` : Change Options attribute VTv4 and later
}
`struct GetAttributeValue` : Get Attribute Command
{
`bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Type attribute
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Width attribute
`bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Height attribute
`bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Background Colour attribute
`bool_t (*BorderColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Border Colour attribute
`bool_t (*KeyCode)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Key Code attribute
`bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T button)` : Get Options attribute VTv4 and later
}

VTv4_ColourMap_T

Structure to hold supported commands for a VTv4 Colour Map object.

Signature
```
typedef struct VTv4_ColourMap_S VTv4_ColourMap_T
```

Members
```
bool_t (*SelectColourMap)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T colour_map) : Select Colour Map command
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T colour_map) : Get Type attribute
}
```

VTv4_Container_T

Structure to hold supported commands for a VTv4 Container object.

Signature
```
typedef struct VTv4_Container_S VTv4_Container_T
```

Members
```
bool_t (*HideShowObject)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T container, ShowHide_Status_T
show_flag) : Hide/Show Object command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T container, ObjectID_T child,
Pixel_T change_x_position, Pixel_T change_y_position,
ObjectPool_ScaleFactor_T scaling_type) : Change Child Location command (Relative
Position)
bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_MessageCallback_T *callback, ObjectID_T container, ObjectID_T
child, Pixel_T new_x_position, Pixel_T new_y_position,
ObjectPool_ScaleFactor_T scaling_type) : Change Child Position command (Absolute
Position)
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T container, Pixel_T new_width, Pixel_T
new_height, ObjectPool_ScaleFactor_T scaling_type) : Change Size command
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T container) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T container) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T container) : Get Height attribute
bool_t (*Hidden)(const VTClient_T *vt_client, const VT_T *vt, const
```

ISOBUS_Callback_T *callback, ObjectID_T container) : Get Hidden attribute
}

VTv4_DataMask_T

Structure to hold supported commands for a VTv4 Data Mask object message.

Signature
typedef struct VTv4_DataMask_S VTv4_DataMask_T

Members
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position) : Change Child Location command (Relative Position)
bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T data_mask, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position) : Change Child Position command (Absolute Position)
bool_t (*ChangeSoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T soft_key_mask) : Change Soft Key Mask command
bool_t (*LockUnlockMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, MaskCommand_T command, ObjectID_T object_id, Time_T timeout) : Lock Unlock Mask command
struct ChangeAttribute : Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, Colour_T background_colour) : Change Background Colour attribute
bool_t (*SoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T soft_key_mask) : Change Soft Key Mask attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Type attribute
bool_t (*BackgroundColour) (const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Background Colour attribute
bool_t (*SoftKeyMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask) : Get Soft Key Mask attribute
}

VTv4_Ellipse_T

Structure to hold supported commands for a VTv4 Ellipse object.

Signature
```
typedef struct VTv4_Ellipse_S VTv4_Ellipse_T
```

Members
`struct ChangeAttribute` : Change Attribute command
{
`bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, ObjectID_T line_attributes)` : Change Line Attributes attribute
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width)` : Change Width attribute
`bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T height)` : Change Height attribute
`bool_t (*EllipseType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse, EllipseType_T ellipse_type)` : Change Ellipse Type attribute
`bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse, Angle_T start)` : Change Start Angle attribute
`bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse, Angle_T end)` : Change End Angle attribute
`bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T rectangle, ObjectID_T fill_attributes)` : Change Fill Attributes attribute
}
`struct GetAttributeValue` : Get Attribute Command
{
`bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Type attribute
`bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Line Attributes attribute
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Width attribute
`bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Height attribute
`bool_t (*EllipseType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Ellipse Type attribute
`bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get Start Angle attribute
`bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse)` : Get End Angle attribute

bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse) : Get Fill Attributes attribute }

VTv4_ExtendedInputAttributes_T

Structure to hold supported commands for a VTv4 Extended Input Attributes object.

Signature
typedef struct VTv4_ExtendedInputAttributes_S VTv4_ExtendedInputAttributes_T

Members
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T extended_input_attributes) : Get Type attribute
bool_t (*ValidationType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T extended_input_attributes) : Get Validation Type attribute
}

VTv4_FillAttributes_T

Structure to hold supported commands for a VTv4 Fill Attributes object.

Signature
typedef struct VTv4_FillAttributes_S VTv4_FillAttributes_T

Members
bool_t (*ChangeFillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes, FillType_T type, Colour_T colour, ObjectID_T pattern_id) : Change Fill Attributes command
struct ChangeAttribute : Change Attribute command
{
bool_t (*FillType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes, FillType_T type) : Change Fill Type attribute
bool_t (*FillColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes, Colour_T colour) : Change Fill Colour attribute
bool_t (*FillPattern)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes, ObjectID_T pattern_id) : Change Fill Pattern attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes) : Get Type attribute
bool_t (*FillType)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T fill_attributes) : Get Fill Type attribute
bool_t (*FillColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes) : Get Fill Colour attribute
bool_t (*FillPattern)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T fill_attributes) : Get Fill Pattern attribute
}

VTv4_FontAttributes_T

Structure to hold supported commands for a VTv4 Font Attributes object.

Signature

```
typedef struct VTv4_FontAttributes_S VTv4_FontAttributes_T
```

Members

bool_t (*ChangeFontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes, Colour_T colour, FontSize_T size, FontType_T type, FontStyle_T style) : Change Font Attributes command
struct ChangeAttribute : Change Attribute command
{
bool_t (*FontColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes, Colour_T colour) : Change Font Colour attribute
bool_t (*FontSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes, FontSize_T size) : Change Font Size attribute
bool_t (*FontType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes, FontType_T type) : Change Font Type attribute
bool_t (*FontStyle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes, FontStyle_T style) : Change Font Style attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes) : Get Type attribute
bool_t (*FontColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes) : Get Font Colour attribute
bool_t (*FontSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes) : Get Font Size attribute
bool_t (*FontType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T font_attributes) : Get Font Type attribute
bool_t (*FontStyle)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T font_attributes) : Get Font Style attribute
}

VTv4_GraphicsContext_T

Structure to hold supported commands for a VTv4 Graphics Context object.

Signature

```
typedef struct VTv4_GraphicsContext_S VTv4_GraphicsContext_T
```

Members

struct GraphicsContext_Command : Graphics Context command
{
bool_t (*SetGraphicsCursor)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T x_position, Pixel_T y_position) : Set Graphics Cursor subcommand
bool_t (*MoveGraphicsCursor)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T x_offset, Pixel_T y_offset) : Move Graphics Cursor subcommand
bool_t (*SetForegroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Colour_T colour) : Set Foreground Colour subcommand
bool_t (*SetBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Colour_T colour) : Set Background Colour subcommand
bool_t (*SetLineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T line_attributes) : Set Line Attributes subcommand
bool_t (*SetFillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T fill_attributes) : Set Fill Attributes subcommand
bool_t (*SetFontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T font_attributes) : Set Font Attributes subcommand
bool_t (*EraseRectangle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T width, Pixel_T height) : Erase Rectangle subcommand
bool_t (*DrawPoint)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T x_offset, Pixel_T y_offset); : Draw Point subcommand
bool_t (*DrawLine)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T x_offset, Pixel_T y_offset) : Draw Line subcommand
bool_t (*DrawRectangle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T width, Pixel_T height) : Draw Rectangle subcommand
bool_t (*DrawClosedEllipse)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T width, Pixel_T height) : Draw Closed Ellipse subcommand

bool_t (*PanViewport)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T viewport_x, Pixel_T viewport_y) : Pan Viewport subcommand
bool_t (*ZoomViewport)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, GraphicsZoom_T zoom, ObjectPool_ScaleFactor_T scaling_type) : Zoom Viewport subcommand
bool_t (*PanAndZoomViewport)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T viewport_x, Pixel_T viewport_y, GraphicsZoom_T zoom, ObjectPool_ScaleFactor_T scaling_type) : Pan and Zoom Viewport subcommand
bool_t (*ChangeViewportSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Viewport Size subcommand
bool_t (*DrawVtObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T object_to_draw) : Draw VT Object subcommand
bool_t (*CopyCanvasToPictureGraphic)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T picture_graphic) : Copy Canvas to Picture Graphic subcommand
bool_t (*CopyViewportToPictureGraphic)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context, ObjectID_T picture_graphic) : Copy Viewport to Picture Graphic subcommand
}
struct ChangeAttribute : Change Attribute command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Type attribute
bool_t (*ViewportWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Viewport Width attribute
bool_t (*ViewportHeight)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Viewport Height attribute
bool_t (*ViewportX)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Viewport X attribute
bool_t (*ViewportY)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Viewport Y attribute
bool_t (*CanvasWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Canvas Width attribute
bool_t (*CanvasHeight)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Canvas Height attribute
bool_t (*ViewportZoom)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Viewport Zoom attribute
bool_t (*GraphicsCursorX)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Graphics Cursor X attribute
bool_t (*GraphicsCursorY)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Graphics Cursor Y attribute
bool_t (*ForegroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Foreground Colour attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Background Colour attribute
bool_t (*FontAttributesObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Font Attributes Object attribute
bool_t (*LineAttributesObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Line Attributes Object attribute
bool_t (*FillAttributesObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Fill Attributes Object attribute
bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Format attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Options attribute
bool_t (*TransparencyColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T graphics_context) : Get Transparency Colour attribute
}

VTv4_InputAttributes_T

Structure to hold supported commands for a VTv4 Input Attributes object.

Signature
```
typedef struct VTv4_InputAttributes_T VTv4_InputAttributes_T
```

Members
bool_t (*ChangeStringValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T input_attributes, Size_T string_length, const char *string) : Change String Value command
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_attributes) : Get Type attribute
bool_t (*ValidationType)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T input_attributes) : Get Validation Type attribute
}

VTv4_InputBoolean_T

Structure to hold supported commands for a VTv4 Input Boolean object.

Signature
typedef struct VTv4_InputBoolean_S VTv4_InputBoolean_T

Members
bool_t (*EnableDisableObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, EnableDisable_Status_T enable_flag) : Enable/Disable Object command
bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Object_SelectionState_T option) : Select Input Object command
bool_t (*ESC)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback) : ESC command
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, NumericValue_T value) : Change Numeric Value command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width, Pixel_T height) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T background_colour) : Change Background Colour attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width) : Change Width attribute
bool_t (*ForegroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T foreground_colour) : Change Foreground Colour attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, ObjectID_T variable) : Change Variable Reference attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Type attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Background Colour attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Width attribute
bool_t (*ForegroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Foreground Colour attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Variable Reference attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Value attribute
bool_t (*Enabled)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean) : Get Enabled attribute
}

VTv4_InputList_T

Structure to hold supported commands for a VTv4 Input List object.

Signature
typedef struct VTv4_InputList_S VTv4_InputList_T

Members
bool_t (*EnableDisableObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, EnableDisable_Status_T enable_flag) : Enable/Disable Object command
bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Object_SelectionState_T option) : Select Input Object command
bool_t (*ESC)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback) : ESC command
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, NumericValue_T value) : Change Numeric Value command
bool_t (*ChangeListItem)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list, ListIndex_T list_index, ObjectID_T new_object) : Change List Item command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width, Pixel_T height) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, Pixel_T height) : Change Height attribute

bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, ObjectID_T variable) : Change Variable Reference attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Height attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Variable Reference attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Value attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_list) : Get Options attribute }

VTv4_InputNumber_T

Structure to hold supported commands for a VTv4 Input Number object.

Signature

```
typedef struct VTv4_InputNumber_S VTv4_InputNumber_T
```

Members

bool_t (*EnableDisableObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, EnableDisable_Status_T enable_flag) : Enable/Disable Object command
bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Object_SelectionState_T option) : Select Input Object command
bool_t (*ESC)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback) : ESC command
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, NumericValue_T value) : Change Numeric Value command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width, Pixel_T height) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width) : Change

Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, Pixel_T height) : Change Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T background_colour) : Change Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, ObjectID_T font_attributes) : Change Font Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumberOptions_T options) : Change Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, ObjectID_T variable) : Change Variable Reference attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumericValue_T minimum) : Change Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumericValue_T maximum) : Change Maximum Value attribute
bool_t (*Offset)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumberOffset_T offset) : Change Offset attribute
bool_t (*Scale)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumberScaleFactor_T scale) : Change Scale attribute
bool_t (*NumberOfDecimals)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumberOfDecimals_T number_of_decimals) : Change Number Of Decimals attribute
bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, NumberFormat_T format) : Change Format attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number, Justification_T justification) : Change Justification attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number) : Get Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number) : Get Background Colour

attribute
`bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Font Attributes attribute
`bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Options attribute
`bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Variable Reference attribute
`bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Minimum Value attribute
`bool_t (*Offset)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Offset attribute
`bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Maximum Value attribute
`bool_t (*Scale)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Scale attribute
`bool_t (*NumberOfDecimals)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Number Of Decimals attribute
`bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Format attribute
`bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Justification attribute
`bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Value attribute
`bool_t (*Options2)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_number)` : Get Options2 attribute
}

VTv4_InputString_T

Structure to hold supported commands for a VTv4 Input String object.

Signature
`typedef struct VTv4_InputString_S VTv4_InputString_T`

Members
`bool_t (*EnableDisableObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, EnableDisable_Status_T enable_flag)` : Enable/Disable Object command
`bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Object_SelectionState_T option)` : Select Input Object command
`bool_t (*ESC)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback)` : ESC command
`bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt,`

const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeStringValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T input_string, Size_T string_length, const char *string) : Change String Value command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_boolean, Pixel_T width, Pixel_T height) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, Pixel_T height) : Change Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, Colour_T background_colour) : Change Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, ObjectID_T font_attributes) : Change Font Attributes attribute
bool_t (*InputAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, ObjectID_T input_attributes) : Change Background Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, StringOptions_T options) : Change Background Colour attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, ObjectID_T variable) : Change Background Colour attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string, Justification_T justification) : Change Background Colour attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Font Attributes attribute
bool_t (*InputAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Input Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Variable Reference attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Justification attribute
bool_t (*Enabled)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T input_string) : Get Enabled attribute
}

VTv4_Key_T

Structure to hold supported commands for a VTv4 Key object.

Signature
typedef struct VTv4_Key_S VTv4_Key_T

Members
bool_t (*SelectInputObject)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key, Object_SelectionState_T option) : Select Input Object command VTv4 and later only
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position) : Change Child Location command (Relative Position)
bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T key, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position) : Change Child Position command (Absolute Position)
struct ChangeAttribute : Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key, Colour_T background_colour) : Change Background Colour attribute
bool_t (*KeyCode)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key, KeyCode_T key_code) : Change Key Code attribute
}
struct GetAttributeValue : Get Attribute Command

{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key) : Get Type attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key) : Get Background Colour attribute
bool_t (*KeyCode)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key) : Get Key Code attribute
}

VTv4_KeyGroup_T

Structure to hold supported commands for a VTv4 Key Group object.

Signature
```
typedef struct VTv4_KeyGroup_S VTv4_KeyGroup_T
```

Members
struct ChangeAttribute : Change Attribute command
{
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key_group, KeyGroupOptions_T options) : Change Options attribute
bool_t (*Name)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key_group, ObjectID_T name) : Change Name attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key_group) : Get Type attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key_group) : Get Options attribute
bool_t (*Name)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T key_group) : Get Name attribute
}

VTv4_Line_T

Structure to hold supported commands for a VTv4 Line object.

Signature
```
typedef struct VTv4_Line_S VTv4_Line_T
```

Members
bool_t (*ChangeEndPoint)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width, Pixel_T height, LineDirection_T direction, ObjectPool_ScaleFactor_T scaling_type) : Change End Point command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width, Pixel_T height,

ObjectPool_ScaleFactor_T scaling_type) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, ObjectID_T line_attributes) : Change Line Attributes attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T height) : Change Height attribute
bool_t (*LineDirection)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, LineDirection_T direction) : Change Line Direction attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line) : Get Type attribute
bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line) : Get Line Attributes attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line) : Get Height attribute
bool_t (*LineDirection)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line) : Get Line Direction attribute
}

VTv4_LineAttributes_T

Structure to hold supported commands for a VTv4 Line Attributes object.

Signature
```
typedef struct VTv4_LineAttributes_S VTv4_LineAttributes_T
```

Members
bool_t (*ChangeLineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes, Colour_T colour, Pixel_T width, LineArt_T line_art, ObjectPool_ScaleFactor_T scaling_type) : Change Line Attributes command
struct ChangeAttribute : Change Attribute command
{
bool_t (*LineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes, Colour_T colour) : Change Line Colour attribute
bool_t (*LineWidth)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T line_attributes, Pixel_T width) : Change Line Width attribute
bool_t (*LineArt)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes, LineArt_T line_art) : Change Line Art attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes) : Get Type attribute
bool_t (*LineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes) : Get Line Colour attribute
bool_t (*LineWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes) : Get Line Width attribute
bool_t (*LineArt)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line_attributes) : Get Line Art attribute
}

VTv4_LinearBarGraph_T

Structure to hold supported commands for a VTv4 Linear Bar Graph object.

Signature
typedef struct VTv4_LinearBarGraph_S VTv4_LinearBarGraph_T

Members
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, NumericValue_T value) : Change Numeric Value attribute
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size attribute
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Pixel_T height) : Change Height attribute
bool_t (*Colour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Colour_T colour) : Change Colour attribute
bool_t (*TargetLineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, Colour_T target_line_colour) : Change Target Line Colour attribute

bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, LinearBarGraphOptions_T options) : Change Options attribute
bool_t (*NumberOfTicks)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, GraphicTicks_T number_of_ticks) : Change Number Of Ticks attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, NumericValue_T minimum_value) : Change Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, NumericValue_T maximum_value) : Change Maximum Value attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, ObjectID_T variable) : Change Variable Reference attribute
bool_t (*TargetValueReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, ObjectID_T target_variable) : Change Target Value Reference attribute
bool_t (*TargetValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, NumericValue_T target_value) : Change Target Value attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Height attribute
bool_t (*Colour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Colour attribute
bool_t (*TargetLineColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Target Line Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Options attribute
bool_t (*NumberOfTicks)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Number Of Ticks attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Maximum Value attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt,

const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Variable Reference attribute
bool_t (*TargetValueReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Target Value Reference attribute
bool_t (*TargetValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Target Value attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph) : Get Value attribute
}

VTv4_Macro_T

Structure to hold supported commands for a VTv4 Macro object.

Signature
typedef struct VTv4_Macro_S VTv4_Macro_T

Members
bool_t (*ExecuteMacro)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, MacroID_T macro) : Execute Macro Command
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T macro) : Get Type attribute
}

VTv4_Meter_T

Structure to hold supported commands for a VTv4 Meter object.

Signature
typedef struct VTv4_Meter_S VTv4_Meter_T

Members
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, NumericValue_T value) : Change Numeric Value attribute
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size attribute
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width) : Change Width attribute
bool_t (*NeedleColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Colour_T needle_colour) :

Change Needle Colour attribute
`bool_t (*BorderColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Colour_T border_colour)` : Change Border Colour attribute
`bool_t (*ArcAndTickColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Colour_T arc_tick_colour)` : Change Arc And Tick Colour attribute
`bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, MeterOptions_T options)` : Change Options attribute
`bool_t (*NumberOfTicks)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, GraphicTicks_T number_of_ticks)` : Change Number Of Ticks attribute
`bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Angle_T start)` : Change Start Angle attribute
`bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Angle_T end)` : Change End Angle attribute
`bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, NumericValue_T minimum_value)` : Change Minimum Value attribute
`bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, NumericValue_T maximum_value)` : Change Maximum Value attribute
`bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, ObjectID_T variable)` : Change Variable Reference attribute
}
`struct GetAttributeValue` : Get Attribute Command
{
`bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Type attribute
`bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Width attribute
`bool_t (*NeedleColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Needle Colour attribute
`bool_t (*BorderColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Border Colour attribute
`bool_t (*ArcAndTickColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Arc And Tick Colour attribute
`bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Options attribute
`bool_t (*NumberOfTicks)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Number Of Ticks attribute
`bool_t (*StartAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter)` : Get Start Angle attribute

bool_t (*EndAngle)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter) : Get End Angle attribute
bool_t (*MinimumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter) : Get Minimum Value attribute
bool_t (*MaximumValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter) : Get Maximum Value attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter) : Get Variable Reference attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter) : Get Value attribute
}

VTv4_NumberVariable_T

Structure to hold supported commands for a VTv4 Number Variable object.

Signature

```
typedef struct VTv4_NumberVariable_S VTv4_NumberVariable_T
```

Members

bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T number_variable, NumericValue_T value)
Change Numeric Value command

struct ChangeAttribute
Change Attribute command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T number_variable)

Change Type attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T number_variable)

Change Value attribute
}

VTv4_ObjectLabelReferenceList_T

Structure to hold supported commands for a VTv4 Object Label Reference List object.

Signature

```
typedef struct VTv4_ObjectLabelReferenceList_S VTv4_ObjectLabelReferenceList_T
```

Members

bool_t (*ChangeObjectLabel)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_label_reference_list,

ObjectID_T string_obj_id, FontType_T font_type, ObjectID_T graphic_obj_id) : Change Object Label command
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_label_reference_list) : Get Type attribute
bool_t (*NumberOfLabelledObjects)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_label_reference_list) : Get Number of Labeled Objects attribute
}

VTv4_ObjectPointer_T

Structure to hold supported commands for a VTv4 Object Pointer object.

Signature
typedef struct VTv4_ObjectPointer_S VTv4_ObjectPointer_T

Members
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_pointer, NumericValue_T value) : Change Numeric Value attribute
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_pointer) : Get Type attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_pointer) : Get Value attribute
}

VTv4_OutputList_T

Structure to hold supported commands for a VTv4 Output List object.

Signature
typedef struct VTv4_OutputList_S VTv4_OutputList_T

Members
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumericValue_T value) : Change Numeric Value command
bool_t (*ChangeListItem)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list, ListIndex_T list_index, ObjectID_T new_object) : Change List Item command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size command
struct ChangeAttribute : Change Attribute command
{

bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T height) : Change Height attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, ObjectID_T variable) : Change Variable Reference attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list) : Get Height attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list) : Get Variable Reference attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_list) : Get Value attribute
}

VTv4_OutputNumber_T

Structure to hold supported commands for a VTv4 Output Number object.

Signature
typedef struct VTv4_OutputNumber_S VTv4_OutputNumber_T

Members
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeNumericValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumericValue_T value) : Change Numeric Value command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T height) : Change Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Colour_T background_colour) : Change Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, ObjectID_T font_attributes) : Change Font Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, StringOptions_T options) : Change Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, ObjectID_T variable) : Change Variable Reference attribute
bool_t (*Offset)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumberOffset_T offset) : Change Offset attribute
bool_t (*Scale)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumberScaleFactor_T scale) : Change Scale attribute
bool_t (*NumberOfDecimals)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumberOfDecimals_T number_of_decimals) : Change Number Of Decimals attribute
bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number, NumberFormat_T format) : Change Format attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Justification_T justification) : Change Justification attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Font Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Variable

Reference attribute
bool_t (*Offset)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Offset attribute
bool_t (*Scale)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Scale attribute
bool_t (*NumberOfDecimals)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Number Of Decimals attribute
bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Format attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Justification attribute
bool_t (*Value)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_number) : Get Value attribute
}

VTv4_OutputString_T

Structure to hold supported commands for a VTv4 Output String object.

Signature
typedef struct VTv4_OutputString_S VTv4_OutputString_T

Members
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeStringValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T output_string, Size_T string_length, const char *string) : Change String Value command
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size command
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Pixel_T height) : Change Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Colour_T background_colour) : Change Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, ObjectID_T font_attributes) : Change Font Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, StringOptions_T

options) : Change Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, ObjectID_T variable) : Change Variable Reference attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string, Justification_T justification) : Change Justification attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Height attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Background Colour attribute
bool_t (*FontAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Font Attributes attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Options attribute
bool_t (*VariableReference)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Variable Reference attribute
bool_t (*Justification)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T output_string) : Get Justification attribute
}

VTv4_PictureGraphic_T

Structure to hold supported commands for a VTv4 Picture Graphic object.

Signature

```
typedef struct VTv4_PictureGraphic_S VTv4_PictureGraphic_T
```

Members

struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T meter, Pixel_T width) : Change Width attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T linear_bar_graph, LinearBarGraphOptions_T options) : Change Options attribute
bool_t (*TransparencyColour)(const VTClient_T *vt_client, const VT_T *vt,

const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic, Colour_T tranparency_colour) : Change Transparency Colour attribute }
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Width attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Options attribute
bool_t (*TransparencyColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Transparency Colour attribute
bool_t (*ActualWidth)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Actual Width attribute
bool_t (*ActualHeight)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Actual Height attribute
bool_t (*Format)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T picture_graphic) : Get Format attribute
}

VTv4_Polygon_T

Structure to hold supported commands for a VTv4 Polygon object.

Signature
```
typedef struct VTv4_Polygon_S VTv4_Polygon_T
```

Members
bool_t (*ChangeSize)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T polygon, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Size attribute
bool_t (*ChangePolygonPoint)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T polygon, PolygonPointIndex_T point_index, Pixel_T x_value, Pixel_T y_value, ObjectPool_ScaleFactor_T scaling_type) : Change Polygon Point attribute
bool_t (*ChangePolygonScale)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T polygon, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T scaling_type) : Change Polygon Scale attribute
struct ChangeAttribute : Change Attribute command
{
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T height) : Change Height

attribute
bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, ObjectID_T line_attributes) : Change Line Attributes attribute
bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T rectangle, ObjectID_T fill_attributes) : Change Fill Attributes attribute
bool_t (*PolygonType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T polygon, PolygonType_T polygon_type) : Change Polygon Type attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse) : Get Type attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse) : Get Height attribute
bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T ellipse) : Get Fill Attributes attribute
bool_t (*PolygonType)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T polygon) : Get Polygon Type attribute
}

VTv4_Rectangle_T

Structure to hold supported commands for a VTv4 Rectangle object.

Signature
typedef struct VTv4_Rectangle_S VTv4_Rectangle_T

Members
struct ChangeAttribute : Change Attribute command
{
bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, ObjectID_T line_attributes) : Change Line Attributes attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T width) : Change Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T line, Pixel_T height) : Change Height attribute
bool_t (*LineSuppression)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T rectangle, LineSuppression_T line_suppression) : Change Line Suppression attribute
bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const

ISOBUS_Callback_T *callback, ObjectID_T rectangle, ObjectID_T
fill_attributes) : Change Fill Attributes attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Type attribute
bool_t (*LineAttributes)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Line Attributes attribute
bool_t (*Width)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Width attribute
bool_t (*Height)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Height attribute
bool_t (*LineSuppression)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Line Suppression attribute
bool_t (*FillAttributes)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T rectangle) : Get Fill Attributes attribute
}

VTv4_SoftKeyMask_T

Structure to hold supported commands for a VTv4 Soft Key Mask object.

Signature
typedef struct VTv4_SoftKeyMask_S VTv4_SoftKeyMask_T

Members
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T soft_key_mask, Colour_T
background_colour) : Change Background Colour command
struct ChangeAttribute : Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T soft_key_mask, Colour_T
background_colour) : Change Background Colour attribute
}
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T soft_key_mask) : Get Type attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, ObjectID_T soft_key_mask) : Get Background Colour
attribute
}

VTv4_StringVariable_T

Structure to hold supported commands for a VTv4 String Variable object.

Signature
typedef struct VTv4_StringVariable_S VTv4_StringVariable_T

Members
bool_t (*ChangeStringValue)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T string_variable, Size_T string_length, const char *string) : Change String Value attribute
struct ChangeAttribute : Change Attribute command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T string_variable) : Change Type attribute
}

VTv4_WindowMask_T

Structure to hold supported commands for a VTv4 Window Mask object.

Signature
typedef struct VTv4_WindowMask_S VTv4_WindowMask_T

Members
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position) : Change Child Location command
bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T window_mask, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position) : Change Child Position command
bool_t (*LockUnlockMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, MaskCommand_T command, ObjectID_T object_id, Time_T timeout) : Lock Unloc Mask command
struct ChangeAttribute : Change Attribute command
{
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask, Colour_T background_colour) : Change Background Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask, WindowMaskOptions_T options) : Change Options attribute
bool_t (*Name)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask, ObjectID_T name) : Change Name attribute
}
struct GetAttributeValue : Get Attribute Command
{

bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask) : Get Type attribute
bool_t (*BackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask) : Get Background Colour attribute
bool_t (*Options)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask) : Get Options attribute
bool_t (*Name)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T window_mask) : Get Name attribute
}

VTv4_WorkingSet_T

Structure to hold supported commands for a VTv4 Working Set object.

Signature
typedef struct VTv4_WorkingSet_S VTv4_WorkingSet_T

Members
bool_t (*ChangeActiveMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set, ObjectID_T new_mask_id) : Change Active Mask command
bool_t (*ChangeBackgroundColour)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set, Colour_T background_colour) : Change Background Colour command
bool_t (*ChangeChildLocation)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set, ObjectID_T child, Pixel_T change_x_position, Pixel_T change_y_position) : Change Child Location command (Relative Position)
bool_t (*ChangeChildPosition)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T working_set, ObjectID_T child, Pixel_T new_x_position, Pixel_T new_y_position) : Change Child Position command (Absolute Position)
struct GetAttributeValue : Get Attribute Command
{
bool_t (*Type)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Type attribute
bool_t (*BackgroundColour) (const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Background Colour attribute
bool_t (*Selectable)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Selectable attribute
bool_t (*ActiveMask)(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T working_set) : Get Active Mask attribute
}

VtChangeActiveMask_T

Structure to hold VT Change Active Mask message data.

Signature
`typedef struct VtChangeActiveMask_S VtChangeActiveMask_T`

Members
`ObjectID_T active_mask_object_id` : Active mask object ID
`ObjectID_T obj_id_with_error` : Object ID containing error
`ObjectID_T parent_obj_id_with_error` : Parent object ID of error object ID
`VT_ChangeActiveMask_ErrorCode_T error_code` : Error code(s): (0 = no error)

VtChangeNumericValue_T

Structure to hold VT Change Numeric Value message data.

Signature

`typedef struct VtChangeNumericValue_S VtChangeNumericValue_T`

Members

`ObjectID_T object_id`
Object ID

`NumericValue_T value`
Value: Size depends on object type. Objects of size 1 byte are found in Byte 5. Objects of size 2 bytes are found in Bytes 5-6. Values greater than 1 byte are transmitted little endian (LSB first).
Unused bytes shall be filled with zero
Input Boolean: 1 byte for TRUE/FALSE
Input Number: 4 bytes for integer input
Input List: 1 byte for list index
Number variable: 4 bytes for integer value

VtChangeSoftKeyMask_T

Structure to hold VT Change Soft Key Mask message data.

Signature
`typedef struct VtChangeSoftKeyMask_S VtChangeSoftKeyMask_T`

Members
`ObjectID_T active_mask_object_id` : Data or Alarm Mask Object ID
`ObjectID_T soft_key_mask_id` : New Soft Key Mask Object ID
`VT_ChangeSoftKeyMask_ErrorCode_T error_code` : Error code(s): (0 = no error)

VtChangeStringValue_T

Structure to hold VT Change String Value message (transport protocol) data.

Signature
`typedef struct VtChangeStringValue_S VtChangeStringValue_T`

Members
`ObjectID_T object_id` : Object ID of the Input String object or String Variable object
`Size_T length` : Total number of bytes in the string to transfer
`Pipe_ReadHandle_T  data` : Entered string value

VTClient_T

Structure to hold Foundation functionality information for an ISOBUS application.

Signature
```
typedef struct VTClient_S VTClient_T
```

Members
`Mutex_T *Mutex` : Pointer to mutex containing priority info used for a VTClient
`Foundation_T *Foundation` : Pointer to the foundation layer
`LanguageCallback_T LanguageCallback` : Language Command Handler
`ActiveVtList_T ActiveVTs` : Active VT List
`struct Transport_MessageHandler_Node_S VTClient_MessageHandler_Node` : Struct for registering a message handler for the VT to ECU message
`struct Transport_MessageHandler_Node_S VTClient_ECUtoVT_Node;` : Struct for registering a message handler for the ECU to VT message
`struct Request_Node_S Request_VTtoECU_Node` : Struct for registering a request handler for the VT to ECU message
`struct Request_Node_S Request_ECUtoVT_Node` : Struct for registering a request handler for the ECU to VT message
`struct Acknowledge_Node_S Acknowledge_ECUtoVT_Node` : Struct for registering a handler to process ACKs to the ECU to VT message
`AuxiliaryFunctionList_T *AuxiliaryFunctionList` : Active Auxiliary Assignments
`AuxiliaryInputList_T *AuxiliaryInputList` : Available Auxiliary Inputs
`VT_T *AuxiliaryVT` : Pointer to Auxilary VT (Function Instance 0)
`struct VtCallbackList` : VT callbacks structure for received messages

VtControlAudioSignalTermination_T

Structure to hold VT Control Audio Signal Termination message data.

Signature
```
typedef struct VtControlAudioSignalTermination_S
VtControlAudioSignalTermination_T
```

Members
`AudioTerminationCause_T  termination_cause` : Audio Termination Cause

VtEsc_T

Structure to hold VT ESC message data.

Signature
```
typedef struct VtEsc_S VtEsc_T
```

Members
`ObjectID_T escape_objectID` : Object ID where input was aborted if no error code
`VT_ESC_ErrorCode_T escape_error_code` : Error code: (0 = no error)

VtOnUserLayoutHideShow_T

Structure to hold VT On User-Layout Hide/Show message data.

Signature
```
typedef struct VtOnUserLayoutHideShow_S VtOnUserLayoutHideShow_T
```

Members
`ObjectID_T mask_1_object_id` : Object ID of Window Mask, or Key Group, Data Mask or Soft Key Mask object
`ObjectID_T mask_2_object_id` : Object ID of Window Mask, or Key Group, Data Mask or Soft Key Mask object or NULL Object ID
`ShowHide_Status_T Mask1_Status` : Status: Bit 0 = State (0 = hidden, 1 = shown)
`ShowHide_Status_T Mask2_Status` : Status: Bit 0 = State (0 = hidden, 1 = shown)

VtSelectInputObject_T

Structure to hold Pointing Event message data.

Signature
```
typedef struct VtSelectInputObject_S VtSelectInputObject_T
```

Members
`ObjectID_T object_id` : Object ID
`Object_SelectionState_T SelectionState` : Selection State

Macros

MAKE_ButtonActivation_Callback_T()

Macro used to initialize a ButtonActivation_Callback_T structure

Signature
```
MAKE_ButtonActivation_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_ChangeActiveMask_Response_Callback_T()

Macro used to initialize a ChangeActiveMask_Response_Callback_T structure

Signature
```
MAKE_ChangeActiveMask_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeAttribute_Response_Callback_T()`

Macro used to initialize a ChangeAttribute_Response_Callback_T structure

Signature
```
MAKE_ChangeAttribute_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to
`attribute_id` : Attribute ID of object attribute to link callback to

`MAKE_ChangeBackgroundColour_Response_Callback_T()`

Macro used to initialize a ChangeBackgroundColour_Response_Callback_T structure

Signature
```
MAKE_ChangeBackgroundColour_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeChildLocation_Response_Callback_T()`

Macro used to initialize a ChangeChildLocation_Response_Callback_T structure

Signature
```
MAKE_ChangeChildLocation_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeChildPosition_Response_Callback_T()`

Macro used to initialize a ChangeChildPosition_Response_Callback_T structure

Signature
```
MAKE_ChangeChildPosition_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeEndPoint_Response_Callback_T()`

Macro used to initialize a ChangeEndPoint_Response_Callback_T structure

Signature
`MAKE_ChangeEndPoint_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeFillAttributes_Response_Callback_T()`

Macro used to initialize a ChangeFillAttributes_Response_Callback_T structure

Signature
`MAKE_ChangeFillAttributes_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeFontAttributes_Response_Callback_T()`

Macro used to initialize a ChangeFontAttributes_Response_Callback_T structure

Signature
`MAKE_ChangeFontAttributes_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeLineAttributes_Response_Callback_T()`

Macro used to initialize a ChangeLineAttributes_Response_Callback_T structure

Signature
`MAKE_ChangeLineAttributes_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeListItem_Response_Callback_T()`

Macro used to initialize a ChangeListItem_Response_Callback_T structure

Signature
`MAKE_ChangeListItem_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_ChangeNumericValue_Response_Callback_T()

Macro used to initialize a ChangeNumericValue_Response_Callback_T structure

Signature

```
MAKE_ChangeNumericValue_Response_Callback_T(function, object_id)
```

Parameters

function
Function to be associated with the callback

object_id
Object ID of object to link callback to

MAKE_ChangePolygonPoint_Response_Callback_T()

Macro used to initialize a ChangePolygonPoint_Response_Callback_T structure

Signature
```
MAKE_ChangePolygonPoint_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_ChangePolygonScale_Response_Callback_T()

Macro used to initialize a ChangePolygonScale_Response_Callback_T structure

Signature
```
MAKE_ChangePolygonScale_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_ChangePriority_Response_Callback_T()

Macro used to initialize a ChangePriority_Response_Callback_T structure

Signature
```
MAKE_ChangePriority_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_ChangeSize_Response_Callback_T()

Macro used to initialize a ChangeSize_Response_Callback_T structure

Signature
`MAKE_ChangeSize_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeSoftKeyMask_Response_Callback_T()`

Macro used to initialize a ChangeSoftKeyMask_Response_Callback_T structure

Signature
`MAKE_ChangeSoftKeyMask_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_ChangeStringValue_Response_Callback_T()`

Macro used to initialize a ChangeStringValue_Response_Callback_T structure

Signature
`MAKE_ChangeStringValue_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_EnableDisableObject_Response_Callback_T()`

Macro used to initialize a EnableDisableObject_Response_Callback_T structure

Signature
`MAKE_EnableDisableObject_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_Esc_Response_Callback_T()`

Macro used to initialize a Esc_Response_Callback_T structure

Signature
`MAKE_Esc_Response_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_Functionalities_T__UniversalTerminal_WorkingSet()

This macro initializes the [Functionalities_T] structure for a Universal Terminal Working Set

Signature
```
MAKE_Functionalities_T__UniversalTerminal_WorkingSet()
```

Parameters
None

MAKE_GetAttributeValue_Response_Callback_T()

Macro used to initialize a GetAttributeValue_Response_Callback_T structure

Signature
```
MAKE_GetAttributeValue_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to
`attribute_id` : Attribute ID of object to link callback to

MAKE_GraphicsContext_Response_Callback_T()

Macro used to initialize a GraphicsContext_Response_Callback_T structure

Signature
```
MAKE_GraphicsContext_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to
`subcommand_id` : Subcommand ID of object to link callback to

MAKE_HideShowObject_Response_Callback_T()

Macro used to initialize a HideShowObject_Response_Callback_T structure

Signature
```
MAKE_HideShowObject_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_SelectColourMap_Response_Callback_T()

Macro used to initialize a SelectColourMap_Response_Callback_T structure

Signature
```
MAKE_SelectColourMap_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_SelectInputObject_Response_Callback_T()`

Macro used to initialize a SelectInputObject_Response_Callback_T structure

Signature
```
MAKE_SelectInputObject_Response_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_SoftKeyActivation_Callback_T()`

Macro used to initialize a SoftKeyActivation_Callback_T structure

Signature
```
MAKE_SoftKeyActivation_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_VtChangeActiveMask_Callback_T()`

Macro used to initialize a VtChangeActiveMask_Callback_T structure

Signature
```
MAKE_VtChangeActiveMask_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_VtChangeNumericValue_Callback_T()`

Macro used to initialize a VtChangeNumericValue_Callback_T structure

Signature

```
MAKE_VtChangeNumericValue_Callback_T(function, object_id)
```

Parameters

`function`
Function to be associated with the callback

`object_id`
Object ID of object to link callback to

MAKE_VtChangeSoftKeyMask_Callback_T()

Macro used to initialize a VtChangeSoftKeyMask_Callback_T structure

Signature
```
MAKE_VtChangeSoftKeyMask_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_VtChangeStringValue_Callback_T()

Macro used to initialize a VtChangeStringValue_Callback_T structure

Signature
```
MAKE_VtChangeStringValue_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_VTClient_T()

This macro initializes the VTClient_T structure

Signature
```
MAKE_VTClient_T(foundation_ptr, vt_array, aux_function_list, aux_input_list,
priority)
```

Parameters
`foundation_ptr` : Pointer to the corresponding Foundation_T structure
`vt_array` : Name of VT_T array
`aux_function_list` : Auxiliary function array
`aux_input_list` : Auxiliary input array
`priority` : Maximum task priority that accesses this structure

MAKE_VtEsc_Callback_T()

Macro used to initialize a VtEsc_Callback_T structure

Signature
```
MAKE_VtEsc_Callback_T(function, object_id)
```

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

MAKE_VtOnUserLayoutHideShow_Callback_T()

Macro used to initialize a VtOnUserLayoutHideShow_Callback_T structure

Signature
`MAKE_VtOnUserLayoutHideShow_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

`MAKE_VtSelectInputObject_Callback_T()`

Macro used to initialize a VtSelectInputObject_Callback_T structure

Signature
`MAKE_VtSelectInputObject_Callback_T(function, object_id)`

Parameters
`function` : Function to be associated with the callback
`object_id` : Object ID of object to link callback to

Functions

`ButtonActivation_Response()`

H.5 Button Activation response (optional)

Sends optional response to Button Activation message

Optionally, in response to Button Activation message

Signature
```
bool_t ButtonActivation_Response(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, const ButtonActivation_T
*message_contents)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeActiveMask_Command()`

F.34 Change Active Mask command

Function sends Vt Command(173) CHANGE ACTIVE MASK to the VT

Sends a destination specific CHANGE ACTIVE MASK command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeActiveMask_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, ObjectID_T
new_mask_id)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`new_mask_id` : New Active Mask Object ID

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeAttribute_Command()`

F.38 Change Attribute command

Function sends Vt Command(175) CHANGE ATTRIBUTE to the VT

Sends a destination specific CHANGE ATTRIBUTE command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeAttribute_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, NumericValue_T
attribute, AttributeID_T attribute_id)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`attribute_id` : Attribute ID (AID)
`attribute` : New value for attribute. Size depends on attribute data type. Values greater than 1 byte are transmitted little endian (LSB first): Boolean: 1 byte for TRUE/FALSE Integer: 1, 2 or 4 bytes as defined in object tables, Float: 4 bytes, Bitmask: 1 byte

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeBackgroundColour_Command()`

Function sends Vt Command(167) CHANGE BACKGROUND COLOUR to the VT

Sends a destination specific CHANGE BACKGROUND COLOUR command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeBackgroundColour_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Colour_T color)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of object to change
`color` : New Background colour

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeChildLocation_Command()`

F.14 Change Child Location command

Function sends Vt Command(165) CHANGE CHILD LOCATION to the VT

Sends a destination specific CHANGE CHILD LOCATION command to the VT. If a callback is provided, it will be called when the VT reply is received **Signature**
```
bool_t ChangeChildLocation_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T parent_obj_id, ObjectID_T
child_obj_id, Pixel_T change_x_position, Pixel_T change_y_position)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`parent_obj_id` : Parent Object ID
`child_obj_id` : Object ID of object to move
`change_x_position` : Relative change in X position
`change_y_position` : Relative change in Y position

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeChildLocation_Command_Scaled()`

F.14 Change Child Location command (Scaled)

Wrapper function for ChangeChildLocation_Command. Determines the type of scaling to be used for the object pool part and calls ChangeChildLocation_Command with the appropriate X and Y values.

Signature
```
bool_t ChangeChildLocation_Command_Scaled(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T parent_obj_id, ObjectID_T child_obj_id, Pixel_T change_x_position, Pixel_T change_y_position, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`parent_obj_id` : Parent Object ID
`child_obj_id` : Object ID of object to move
`change_x_position` : Relative change in X position
`change_y_position` : Relative change in Y position
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if ChangeChildLocation_Command_Scaled was successful
: `FALSE` if ChangeChildLocation_Command_Scaled was not successful

`ChangeChildLocation_Command_Scaled_DataMask()`

F.14 Change Child Location command (Soft Key Mask)

Softkey mask wrapper function for ChangeChildLocation_Command_Scaled. Automatically applies softkey mask scaling for the ChangeChildLocation_Command.

Signature
```
bool_t ChangeChildLocation_Command_Scaled_DataMask(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T parent_object_id, ObjectID_T child_object_id, Pixel_T change_x_position, Pixel_T change_y_position)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`parent_object_id` : Parent Object ID
`child_object_id` : Object ID of object to move
`change_x_position` : Relative change in X position
`change_y_position` : Relative change in Y position

Returns
`bool_t`

: TRUE if ChangeChildLocation_Command_Scaled_SoftKeyMask was successful
: FALSE if ChangeChildLocation_Command_Scaled_SoftKeyMask was not successful

ChangeChildLocation_Command_Scaled_SoftKeyMask()

F.16 Change Child Position command (Data Mask)

Datamask wrapper function for ChangeChildPosition_Command_Scaled. Automatically applies datamask scaling for the ChangeChildPosition_Command.

Signature

```
bool_t ChangeChildLocation_Command_Scaled_SoftKeyMask(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
parent_object_id, ObjectID_T child_object_id, Pixel_T change_x_position,
Pixel_T change_y_position)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
parent_object_id : Parent Object ID
child_object_id : Object ID of object to move
new_x_position : New X position relative to the top left corner of parent object
new_y_position : New Y position relative to the top left corner of parent object

Returns

bool_t
: TRUE if ChangeChildPosition_Command_Scaled_DataMask was successful
: FALSE if ChangeChildPosition_Command_Scaled_DataMask was not successful

ChangeChildPosition_Command()

F.16 Change Child Position command

Function sends Vt Command(180) CHANGE CHILD POSITION to the VT

Sends a destination specific CHANGE CHILD POSITION command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeChildPosition_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T parent_obj_id,
ObjectID_T object_id, Pixel_T new_x_position, Pixel_T new_y_position)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
parent_obj_id : Parent Object ID
object_id : Object ID of object to move

new_x_position : New X position relative to the top left corner of parent object
new_y_position : New Y position relative to the top left corner of parent object

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

ChangeChildPosition_Command_Scaled()

F.16 Change Child Position command (Scaled)

Wrapper function for ChangeChildPosition_Command. Determines the type of scaling to be used for the object pool part and calls ChangeChildPosition_Command with the appropriate X and Y values.

Signature
```
bool_t ChangeChildPosition_Command_Scaled(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T parent_obj_id, ObjectID_T object_id, Pixel_T new_x_position, Pixel_T new_y_position, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
parent_obj_id : Parent Object ID
object_id : Object ID of object to move
new_x_position : New X position relative to the top left corner of parent object
new_y_position : New Y position relative to the top left corner of parent object
scaling_type : Indicates how to scale this object pool part

Returns
bool_t
: TRUE if ChangeChildPosition_Command_Scaled was successful
: FALSE if ChangeChildPosition_Command_Scaled was not successful

ChangeChildPosition_Command_Scaled_SoftKeyMask()

F.16 Change Child Position command (Soft Key Mask)

Softkey mask wrapper function for ChangeChildPosition_Command_Scaled. Automatically applies softkey mask scaling for the ChangeChildPosition_Command.

Signature
```
bool_t ChangeChildPosition_Command_Scaled_SoftKeyMask(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_MessageCallback_T *callback, ObjectID_T parent_object_id, ObjectID_T child_object_id, Pixel_T new_x_position, Pixel_T new_y_position)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`parent_object_id` : Parent Object ID
`child_object_id` : Object ID of object to move
`new_x_position` : New X position relative to the top left corner of parent object
`new_y_position` : New Y position relative to the top left corner of parent object

Returns
`bool_t`
: `TRUE` if ChangeChildPosition_Command_Scaled_SoftKeyMask was successful
: `FALSE` if ChangeChildPosition_Command_Scaled_SoftKeyMask was not successful

`ChangeEndPoint_Command()`

F.26 Change End Point command

Function sends Vt Command(169) CHANGE END POINT to the VT

Sends a destination specific CHANGE END POINT command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeEndPoint_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T width,
Pixel_T height, LineDirection_T direction)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`width` : Width in pixels
`height` : Height in pixels
`direction` : Line Direction (refer to Line object attributes)

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeEndPoint_Command_Scaled()`

F.26 Change End Point command (Scaled)

Wrapper function for ChangeEndPoint_Command. Determines the type of scaling to be used for changing a line endpoint and calls ChangeEndPoint_Command with the appropriate width and height values.

Signature
```
bool_t ChangeEndPoint_Command_Scaled(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T width,
Pixel_T height, LineDirection_T direction, ObjectPool_ScaleFactor_T
scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`width` : Width in pixels
`height` : Height in pixels
`direction` : Line Direction (refer to Line object attributes)
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if ChangeEndPoint_Command_Scaled was successful
: `FALSE` if ChangeEndPoint_Command_Scaled was not successful

`ChangeFillAttributes_Command()`

F.32 Change Fill Attributes command

Function sends Vt Command(172) CHANGE FILL ATTRIBUTES to the VT

Sends a destination specific CHANGE FILL ATTRIBUTES command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeFillAttributes_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, FillType_T
type, Colour_T color, ObjectID_T pattern_id)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`type` : Fill Type
`color` : Fill Colour
`pattern_id` : Fill Pattern object ID

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

ChangeFontAttributes_Command()

F.28 Change Font Attributes command

Function sends Vt Command(170) CHANGE FONT ATTRIBUTES to the VT

Sends a destination specific CHANGE FONT ATTRIBUTES command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeFontAttributes_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Colour_T color,
FontSize_T size, FontType_T type, FontStyle_T style)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`color` : Font colour
`size` : Font size
`type` : Font type
`style` : Font style

Returns

`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

ChangeLineAttributes_Command()

F.30 Change Line Attributes command

Function sends Vt Command(171) CHANGE LINE ATTRIBUTES to the VT

Sends a destination specific CHANGE LINE ATTRIBUTES command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeLineAttributes_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Colour_T color,
Pixel_T width, LineArt_T line_art)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of object to change
`color` : Line Colour

`width` : Line Width
`line_art` : Line Art

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeLineAttributes_Command_Scaled()`

F.30 Change Line Attributes command (Scaled)

Wrapper function for ChangeLineAttributes_Command. Determines the type of scaling to be used for changing line attributes and calls ChangeFontAttributes_Command with the appropriate size value.

Signature
```
bool_t ChangeLineAttributes_Command_Scaled(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Colour_T color, Pixel_T width, LineArt_T line_art, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of object to change
`color` : Line Colour
`width` : Line Width
`line_art` : Line Art
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if ChangeLineAttributes_Command_Scaled was successful
: `FALSE` if ChangeLinteAttributes_Command_Scaled was not successful

`ChangeListItem_Command()`

F.42 Change List Item command

Function sends Vt Command(177) CHANGE LIST ITEM to the VT

Sends a destination specific CHANGE LIST ITEM command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeListItem_Command(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, ListIndex_T list_index, ObjectID_T new_object_id)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`list_index` : List Index (items are numbered 0-n)
`new_object_id` : New object ID or NULL_OBJECT_ID to set empty item

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeNumericValue_Command()`

F.22 Change Numeric Value Command

Sends the Change Numeric Value Command message to the VT, which tells the VT to change the numeric value of a given number-oriented object; most notably for Number Variable objects, but also for Object Pointer objects and raw values of input/output objects.

Signature

```
bool_t ChangeNumericValue_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, NumericValue_T
value)
```

Parameters

`vt_client`
Pointer to the application's VTClient data structure

`vt`
Pointer to the application's active data structure

`callback`
Callback when message is sent

`object_id`
Object ID of object to change

`value`
New value for value attribute. Size depends on object type. Objects of size 1 byte are found in Byte 5. Objects of size 2 bytes are found in Bytes 5-6. Values greater than 1 byte are transmitted little endian (LSB first). Unused bytes shall be filled with zero.
Boolean input object: 1 byte for TRUE/FALSE
Number input object: 4 bytes for integer input
List input object: 1 byte for list index
List output object: 1 byte for list index
Number output object: 4 bytes for integer output

Meter: 2 bytes for integer value
Linear bar graph: 2 bytes for integer value
Arched bar graph: 2 bytes for integer value
Number variable: 4 bytes for integer value
Object pointer: 2 bytes for Object ID

Returns

bool_t
TRUE if the message was queued to be sent
FALSE if the message was not queued

ChangeObjectLabel_Callback_Register()

Registers the change object label callback function

Signature

```
void ChangeObjectLabel_Callback_Register(VTClient_T *vt_client,
void(*ChangeObjectLabel_Response)(VTClient_T *vt_client, const VT_T *vt,
const ChangeObjectLabel_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs ChangeObjectLabel_Response

Function pointer to the desired callback function

Returns

(void)

ChangeObjectLabel_Command()

F.50 Change Object Label command

Function sends Vt Command(181) CHANGE OBJECT LABEL to the VT

Sends a destination specific CHANGE OBJECT LABEL command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeObjectLabel_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, ObjectID_T
string_obj_id, FontType_T font_type, ObjectID_T graphic_obj_id)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
object_id : Object ID of object to associate label with

`string_obj_id` : Object ID of a String Variable object that contains the label string (32 characters maximum) or NULL_OBJECT_ID if no text is supplied
`font_type` : Font type (ignored if String Variable object reference is NULL Object ID or the string contains a WideString)
`graphic_obj_id` : Object ID of an object to be used as a graphic representation of the object label or NULL_OBJECT_ID if no designator supplied. When the VT draws this object it shall be clipped to the size of a Soft Key designator

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangePolygonPoint_Command()`

F.52 Change Polygon Point command

Function sends Vt Command(182) CHANGE POLYGON POINT to the VT. Sends a destination specific CHANGE POLYGON POINT command to the VT. If a callback is provided, it will be called when the VT reply is received.

Signature
```
bool_t ChangePolygonPoint_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
PolygonPointIndex_T point_index, Pixel_T new_x_value, Pixel_T new_y_value)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of the Polygon object to change
`point_index` : Point index of the point to replace
`new_x_value` : New X value of a point relative to the top left corner of the polygon
`new_y_value` : New Y value of a point relative to the top left corner of the polygon

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangePolygonPoint_Command_Scaled()`

F.52 Change Polygon Point command (Scaled)

Wrapper function for ChangePolygonPoint_Command. Determines the type of scaling to be used when changing the start point of a polygon and calls ChangePolygonPoint_Command with the appropriate (x,y) values.

Signature
```
bool_t ChangePolygonPoint_Command_Scaled(const VTClient_T *vt_client, const
```

```
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
PolygonPointIndex_T point_index, Pixel_T new_x_value, Pixel_T new_y_value,
ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of the Polygon object to change
`point_index` : Point index of the point to replace
`new_x_value` : New X value of a point relative to the top left corner of the polygon
`new_y_value` : New Y value of a point relative to the top left corner of the polygon
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if ChangePolygonPoint_Command_Scaled was successful
: `FALSE` if ChangePolygonPoint_Command_Scaled was not successful

`ChangePolygonScale_Command()`

F.54 Change Polygon Scale command

Function sends Vt Command(183) CHANGE POLYGON SCALE to the VT. Sends a destination specific CHANGE POLYGON SCALE command to the VT. If a callback is provided, it will be called when the VT reply is received.

Signature
```
bool_t ChangePolygonScale_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T
new_width, Pixel_T new_height)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Polygon object to scale
`new_width` : New width attribute
`new_height` : New height attribute

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangePolygonScale_Command_Scaled()`

F.54 Change Polygon Scale command (Scaled)

Wrapper function for ChangePolygonScale_Command. Determines the type of scaling to be used when changing the scale of a polygon and calls ChangePolygonScale_Command with the appropriate width and height values.

Signature
```
bool_t ChangePolygonScale_Command_Scaled(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T new_width, Pixel_T new_height, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Polygon object to scale
`new_width` : New width attribute
`new_height` : New height attribute
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if ChangePolygonScale_Command_Scaled was successful
: `FALSE` if ChangePolygonScale_Command_Scaled was not successful

`ChangePriority_Command()`

F.40 Change Priority command

Function sends Vt Command(176) CHANGE PRIORITY to the VT

Sends a destination specific CHANGE PRIORITY command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangePriority_Command(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, AlarmPriority_T priority)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of a Line object to change
`priority` : New priority

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

ChangeSize_Command()

Function sends Vt Command(166) CHANGE SIZE to the VT

Sends a destination specific CHANGE SIZE command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature
```
bool_t ChangeSize_Command(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T new_width, Pixel_T new_height)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of the object to change size
`new_width` : New width
`new_height` : New height

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

ChangeSize_Command_Scaled()

F.18 Change Size command (Scaled)

Wrapper function for ChangeSize_Command. Determines the type of scaling to be used for changing an object pool part size and calls ChangeSize_Command with the appropriate width and height values.

Signature
```
bool_t ChangeSize_Command_Scaled(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T new_width, Pixel_T new_height, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of the object to change size
`new_width` : New width
`new_height` : New height
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`

: TRUE if ChangeSize_Command_Scaled was successful
: FALSE if ChangeSize_Command_Scaled was not successful

ChangeSize_Command_Scaled_DataMask()

F.18 Change Size command (Data Mask)

Datamask wrapper function for ChangeSize_Command_Scaled. Automatically applies datamask scaling for the ChangeSize_Command.

Signature

```
bool_t ChangeSize_Command_Scaled_DataMask(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T new_width, Pixel_T new_height)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
object_id : Object ID of the object to change size
new_width : New width
new_height : New height

Returns

bool_t
: TRUE if ChangeSize_Command_Scaled_DataMask was successful
: FALSE if ChangeSize_Command_Scaled_DataMask was not successful

ChangeSoftKeyMask_Command()

F.36 Change Soft Key Mask command

Function sends Vt Command(174) CHANGE ACTIVE MASK to the VT

Sends a destination specific CHANGE ACTIVE MASK command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeSoftKeyMask_Command(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T mask_object_id, ObjectID_T new_soft_key_id, MaskType_T mask_type)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
mask_type : Mask Type (1 = Data, 2 = Alarm)
mask_object_id : Data or Alarm Mask Object ID
new_soft_key_id : New Soft Key Mask Object ID

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`ChangeSoftKeyMask_AlarmMask()`

F.14 Change Child Location command (Data Mask)

Datamask wrapper function for ChangeChildLocation_Command_Scaled. Automatically applies datamask scaling for the ChangeChildLocation_Command.

Signature
```
bool_t ChangeSoftKeyMask_AlarmMask(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T alarm_mask, ObjectID_T
soft_key_mask)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`parent_object_id` : Parent Object ID
`child_object_id` : Object ID of object to move
`change_x_position` : Relative change in X position
`change_y_position` : Relative change in Y position

Returns
`bool_t`
: `TRUE` if ChangeChildLocation_Command_Scaled_DataMask was successful
: `FALSE` if ChangeChildLocation_Command_Scaled_DataMask was not successful

`ChangeSoftKeyMask_DataMask()`

F.36 Change Soft Key Mask command (Data Mask). Automatically applies the data mask parameter for the ChangeSoftKeyMask_Command.

Signature
```
bool_t ChangeSoftKeyMask_DataMask(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T data_mask, ObjectID_T
soft_key_mask)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`data_mask` : Data Mask Object ID
`soft_key_mask` : Soft Key Mask Object ID

Returns
`bool_t`

: TRUE if the message was queued to be sent
: FALSE if the message was not queued

ChangeStringValue_Command()

F.24 Change String Value command (transport protocol)

Function sends Vt Command(179) CHANGE STRING VALUE to the VT

Sends a destination specific CHANGE STRING VALUE command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ChangeStringValue_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_MessageCallback_T *callback, ObjectID_T object_id, Size_T
string_length, const char *string)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID of the object to change
`string_length` : Total number of bytes in the string to transfer
`string` : Pointer to a string

Returns

`bool_t`
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

ControlAudioSignal_Callback_Register()

Registers the control audio signal callback function

Signature

```
void ControlAudioSignal_Callback_Register(VTClient_T *vt_client,
void(*ControlAudioSignal_Response)(VTClient_T *vt_client, const VT_T *vt,
const ControlAudioSignal_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `ControlAudioSignal_Response`

Function pointer to the desired callback function

Returns

(void)

ControlAudioSignal_Command()

F.10 Control Audio Signal command

Function sends Vt Command(163) CONTROL AUDIO SIGNAL to the VT

Sends a destination specific CONTROL AUDIO SIGNAL command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ControlAudioSignal_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, AudioSignalActivation_T activations,
Frequency_T frequency, Time_T on_time, Time_T off_time)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`activations` : Activations 0 = Terminates any audio in process from the originating ECU (Frequency and Duration values are ignored). 1-255 = Number of Audio Activations
`frequency` : Frequency in Hz. If the Frequency specified is outside the VT capabilities for production of sound (also applies to non-multiple frequency devices) then the VT limits the frequency to the reproducible range
`on_time` : On-time duration in ms. If the duration specified is less than the VT capabilities for timing, the VT shall time the audio to the VT's smallest controlled value
`off_time` : Off-time duration in ms. If the duration specified is less than the VT capabilities for timing, the VT shall time the audio to the VT's smallest controlled value

Returns

`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

DeleteObjectPool_Command()

F.44 Delete Object Pool command

Sends Delete Object Pool command

Signature

```
bool_t DeleteObjectPool_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns

`bool_t`

: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`DeleteVersion_Callback_Register()`

Registers the delete version callback function

Signature

```
void DeleteVersion_Callback_Register(VTClient_T *vt_client,
void(*DeleteVersion_Response)(VTClient_T *vt_client, const VT_T *vt, const
DeleteVersion_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `DeleteVersion_Response`

Function pointer to the desired callback function

Returns

(void)

`DeleteVersion_Command()`

Sends a command to the VT to delete the working set's object pool with supplied version label. The pool will be deleted from the VT's non-volatile memory (ROM) -- it won't necessarily be deleted from the VT's volatile memory (RAM).

If the `version` array contains all spaces, then this command instructs the VT to delete the last saved version of the working set's object pool from non-volatile memory.

The user can know the delete version command was successful by listening for the Delete Version Response message from the VT. A callback for this message can be registered using the `DeleteVersion_Callback_Register()` function.

Signature

```
bool_t DeleteVersion_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, const char *version)
```

Parameters

`vt_client`
VTClient structure containing all active VTs

`vt`
VT instance

`callback`
Callback when message is sent

version
Version label of the pool version to delete. Must be exactly 7 characters, with any non-used characters at the end filledin with space characters.

Returns

bool_t
TRUE Message was sent to the VT.
FALSE Message was not sent

EnableDisableObject_Command()

F.4 Enable/Disable Object command

Function sends Vt Command(161) ENABLE/DISABLE OBJECT to the VT

Sends a destination specific ENABLE/DISABLE OBJECT command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t EnableDisableObject_Command(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
EnableDisable_Status_T enable_flag)
```

Parameters

vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
object_id : Object ID
enable_flag : 0 = Disable, 1 = Enable

Returns

bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

Esc_Command()

F.8 ESC command

Function sends Vt Command(146) ECU ESC to the VT

Sends a destination specific ECU ESC command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t Esc_Command(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback)
```

Parameters

vt_client : Pointer to the application's VTClient data structure

vt : Pointer to the application's active data structure
callback : Callback when message is sent

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

ExecuteMacro_Callback_Register()

Registers the execute macro callback function

Signature

```
void ExecuteMacro_Callback_Register(VTClient_T *vt_client,
void(*ExecuteMacro_Response)(VTClient_T *vt_client, const VT_T *vt, const
ExecuteMacro_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs ExecuteMacro_Response

Function pointer to the desired callback function

Returns

(void)

ExecuteMacro_Command()

F.48 Execute Macro command

Function sends Vt Command(190) EXECUTE MACRO to the VT

Sends a destination specific EXECUTE MACRO command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t ExecuteMacro_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, MacroID_T macro)
```

Parameters
vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
macro : Object ID of Macro object

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GetAttributeValue_Message()

F.58 Get Attribute Value message

Function sends Vt Command(185) GET ATTRIBUTE VALUE to the VT.

Sends a destination specific GET ATTRIBUTE VALUE command to the VT. If a callback is provided, it will be called when the VT reply is received.

Signature

```
bool_t GetAttributeValue_Message(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, AttributeID_T
attribute_id)
```

Parameters

`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`attribute_id` : Attribute ID of the Object

Returns

`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

GetHardware_Callback_Register()

Registers the get hardware callback function

Signature

```
void GetHardware_Callback_Register(VTClient_T *vt_client,
void(*GetHardware_Response)(VTClient_T *vt_client, const VT_T *vt, const
GetHardware_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `GetHardware_Response`

Function pointer to the desired callback function

Returns

(void)

GetHardware_Message()

D.8 Get Hardware message

The Get Hardware message informs the Working Set as to the hardware design of the VT

Signature
`bool_t GetHardware_Message(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback)`

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GetMemory_Message()`

D.2 Get Memory message

The Get Memory message allows the Working Set to determine if the VT is out of memory and also determines the VT version

Signature

`bool_t GetMemory_Message(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, Size_T memory_required)`

Parameters

`vt_client`
Pointer to the application's VTClient data structure

`vt`
Pointer to the application's active data structure

`callback`
Callback when message is sent

`memory_required`
Number of bytes in object pool

Returns

`bool_t`
`TRUE` if the message was sent
`FALSE` if the message was not sent

`GetMemory_Response_Callback_Register()`

Registers the get memory response callback function

Signature

```
void GetMemory_Response_Callback_Register(VTClient_T *vt_client,
void(*GetMemory_Response)(VTClient_T *vt_client, const VT_T *vt, const
GetMemory_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `GetMemory_Response`

Function pointer to the desired callback function

Returns

(void)

GetNumberOfSoftKeys_Callback_Register()

Registers the get number of soft keys callback function

Signature

```
void GetNumberOfSoftKeys_Callback_Register(VTClient_T *vt_client,
void(*GetNumberOfSoftKeys_Response)(VTClient_T *vt_client, const VT_T *vt,
const GetSoftKeys_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `GetNumberOfSoftKeys_Response`

Function pointer to the desired callback function

Returns

(void)

GetNumberOfSoftKeys_Message()

D.4 Get Number of Soft Keys message

The Get Number of Soft Keys message supplies the Working Set with the available divisions of the X and Y axes for Soft Key descriptors, the available virtual Soft Keys and the number of physical Soft Keys

Signature
```
bool_t GetNumberOfSoftKeys_Message(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GetSupportedObjects_Callback_Register()`

Registers the get supported objects callback function

Signature

```
void GetSupportedObjects_Callback_Register(VTClient_T *vt_client,
void(*GetSupportedObjects_Response)(VTClient_T *vt_client, const VT_T *vt,
void *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `GetSupportedObjects_Response`

Function pointer to the desired callback function

Returns

(void)

`GetSupportedObjects_Message()`

D.14 Get Supported Objects message

The Get Supported Objects message is used by the Working Set to get the list of all object types supported by the VT

Signature
```
bool_t GetSupportedObjects_Message(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback)
```

Parameters
`vt_client` : VTClient structure containing all active VTs
`vt` : VT instance
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GetSupportedWideChars_Callback_Register()`

Registers the get supported whide chars callback function

Signature

```
void GetSupportedWideChars_Callback_Register(VTClient_T *vt_client,
void(*GetSupportedWideChars_Response)(VTClient_T *vt_client, const VT_T *vt,
void *))
```

Parameters

vt_client
VTClient structure containing all active VTs `GetSupportedWideChars_Response`

Function pointer to the desired callback function

Returns

(void)

GetSupportedWideChars_Message()

D.10 Get Supported WideChars message

The Get Suppored WideChars message supplies the Working Set with a list of the WideChars supported by the VT

Signature
```
bool_t GetSupportedWideChars_Message(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, WideChar_CodePlane_T codeplane,
WideChar_T first_wide_char, WideChar_T last_wide_char)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`codeplane` : Code Plane of the wide char range requested
`first_wide_char` : First WideChar in inquiry range
`last_wide_char` : Last WideChar in inquiry range

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

GetTextFontData_Callback_Register()

Registers the get text font data callback function

Signature

```
void GetTextFontData_Callback_Register(VTClient_T *vt_client,
void(*GetTextFontData_Response)(VTClient_T *vt_client, const VT_T *vt, const
GetTextFont_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `GetTextFontData_Response`

Function pointer to the desired callback function

Returns

(void)

`GetTextFontData_Message()`

D.6 Get Text Font Data message

The Get Text Font Data message provides the Working Set with the characteristics of fonts, type sizes, type attributes and colour capabilities

Signature
```
bool_t GetTextFontData_Message(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GetVersions_Callback_Register()`

Registers the get versions callback function

Signature

```
void GetVersions_Callback_Register(VTClient_T *vt_client,
void(*GetVersions_Response)(VTClient_T *vt_client, const VT_T *vt,
GetVersions_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `GetVersions_Response`

Function pointer to the desired callback function

Returns

(void)

GetVersions_Message()

E.2 Get Versions message

Requests list of object pool versions stored on the VT

The Get Versions message allows the Working Set to query the VT for existing version labels associated with the requesting Working Set

Signature

```
bool_t GetVersions_Message(const VTClient_T *vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback)
```

Parameters

`vt_client` : VTClient structure containing all active VTs
`vt` : VT instance
`callback` : Callback when message is sent

Returns

`bool_t`
: `TRUE` Message was sent
: `FALSE` Message was not sent (try again later)

GetWindowMaskData_Callback_Register()

Registers the get window mask data callback function

Signature

```
void GetWindowMaskData_Callback_Register(VTClient_T *vt_client, void(*GetWindowMaskData_Response)(VTClient_T *vt_client, const VT_T *vt, const GetWindowMask_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `GetWindowMaskData_Response`

Function pointer to the desired callback function

Returns

(void)

GetWindowMaskData_Message()

D.12 Get Window Mask Data message

The Get Window Mask Data message provides the Working Set with the background colour of User-Layout Data Mask and background colour of the Key Cells on a User-Layout Soft Key Mask

Signature
```
bool_t GetWindowMaskData_Message(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_ChangeViewportSize_Command()`

F.56 Graphics Context command (Change Viewport Size)

Changes the size of the viewport (Cursor not moved)

This command changes the size of the viewport and can be compared to the normal Change Size command.

Note: The size of the object (i.e. the memory used) cannot be changed. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_ChangeViewportSize_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Pixel_T width, Pixel_T height)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`width` : Viewport width
`height` : Viewport height

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_ChangeViewportSize_Command_Scaled()`

F.56 Subcommand 17: Change Viewport Size (Scaled)

Wrapper function for GraphicsContext_ChangeViewportSize_Command. Determines the type of scaling to be used when changing the viewport size and calls

GraphicsContext_ChangeViewportSize_Command with the appropriate width and height values.

Signature

```
bool_t GraphicsContext_ChangeViewportSize_Command_Scaled(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Pixel_T width, Pixel_T height, ObjectPool_ScaleFactor_T
scaling_type)
```

Parameters

`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`width` : Width of the viewport
`height` : Height of the viewport
`scaling_type` : Indicates how to scale this object pool part

Returns

`bool_t`
: `TRUE` if GraphicsContext_ChangeViewportSize_Command_Scaled was successful
: `FALSE` if GraphicsContext_ChangeViewportSize_Command_Scaled was not successful

`GraphicsContext_CopyCanvasToPictureGraphic_Command()`

F.56 Graphics Context command (Copy Canvas to Picture Graphic)

Copies Canvas to Picture Graphic object (Cursor not moved)

Signature

```
bool_t GraphicsContext_CopyCanvasToPictureGraphic_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, ObjectID_T picture_graphic)
```

Parameters

`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`picture_graphic` : Object in which to store copy of canvas

Returns

`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_CopyViewportToPictureGraphic_Command()`

F.56 Graphics Context command (Copy Viewport to Picture Graphic)

Copies Viewport to Picture Graphic object (Cursor not moved)

Signature
```
bool_t GraphicsContext_CopyViewportToPictureGraphic_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, ObjectID_T picture_graphic)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`picture_graphic` : Object in which to store copy of canvas

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawClosedEllipse_Command()`

F.56 Graphics Context command (Draw Closed Ellipse)

Draws closed ellipse using line/fill attributes (Cursor moved to bottom right)

Signature
```
bool_t GraphicsContext_DrawClosedEllipse_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
Pixel_T width, Pixel_T height)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`width` : Ellipse width
`height` : Ellipse height

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawLine_Command()`

F.56 Graphics Context command (Draw Line)

Draws line from cursor to point (moves cursor to end point)

Signature
```
bool_t GraphicsContext_DrawLine_Command(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T
x_offset, Pixel_T y_offset)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`x_offset` : Relative x location
`y_offset` : Relative y location

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawPoint_Command()`

F.56 Graphics Context command (Draw Point)

Sets pixel to foreground colour (moves cursor to point)

Signature
```
bool_t GraphicsContext_DrawPoint_Command(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T
x_offset, Pixel_T y_offset)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`x_offset` : Relative x location
`y_offset` : Relative y location

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawPolygon_Command()`

F.56 Graphics Context command (Draw Polygon)

Draws a polygon using line/fill attributes (Cursor moved to last point)

Signature
```
bool_t GraphicsContext_DrawPolygon_Command(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with

`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawRectangle_Command()`

F.56 Graphics Context command (Draw Rectangle)

Draws rectangle using line/fill attributes (Cursor moved to bottom right)

Signature
```
bool_t GraphicsContext_DrawRectangle_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
Pixel_T width, Pixel_T height)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`width` : Rectangle width
`height` : Rectangle height

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_DrawText_Command()`

F.56 Graphics Context command (Draw Text)

Draws text using Font Attributes (Cursor moved to bottom right)

Signature
```
bool_t GraphicsContext_DrawText_Command(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify

Returns
`bool_t`

: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_DrawVtObject_Command()

F.56 Graphics Context command (Draw VT Object)

Draws arbitrary object (Cursor moved to bottom right)

Signature
```
bool_t GraphicsContext_DrawVtObject_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
ObjectID_T object_to_draw)
```

Parameters
vt_client : VTClient data structure
vt : VT to interact with
callback : Callback when message is sent
object_id : Object ID of the Graphics Context object to modify
object_to_draw : Object ID of the object to draw

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_EraseRectangle_Command()

F.56 Graphics Context command (Erase Rectangle)

Fill rectangle with background color (Cursor moved to bottom right)

Signature
```
bool_t GraphicsContext_EraseRectangle_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
Pixel_T width, Pixel_T height)
```

Parameters
vt_client : VTClient data structure
vt : VT to interact with
callback : Callback when message is sent
object_id : Object ID of the Graphics Context object to modify
width : Rectangle width
height : Rectangle height

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_MoveGraphicsCursor_Command()

F.56 Graphics Context command (Move Graphics Cursor)

This command alters the graphics cursor X/Y attributes of the object by moving it relative to its current position.

Signature
```
bool_t GraphicsContext_MoveGraphicsCursor_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Pixel_T x_offset, Pixel_T y_offset)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`x_offset` : Cursor X offset
`y_offset` : Cursor Y offset

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

GraphicsContext_PanAndZoomViewport_Command()

F.56 Graphics Context command (Pan and Zoom Viewport)

Modifies viewport location and magnification (Cursor not moved)

This command allows both panning and zooming at the same time combining commands 14 and 15.

Signature
```
bool_t GraphicsContext_PanAndZoomViewport_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Pixel_T viewport_x, Pixel_T viewport_y, GraphicsZoom_T zoom)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`viewport_x` : Relative x location
`viewport_y` : Relative y location
`zoom` : Magnification

Returns
`bool_t`

: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_PanAndZoomViewport_Command_Scaled()

F.56 Subcommand 16: Pan and Zoom Viewport (Scaled)

Wrapper function for GraphicsContext_PanAndZoomViewport_Command. Determines the type of scaling to be used when panning and zooming the viewport and calls GraphicsContext_PanAndZoomViewport_Command with the appropriate zoom value.

Signature

```
bool_t GraphicsContext_PanAndZoomViewport_Command_Scaled(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Pixel_T viewport_x, Pixel_T viewport_y, GraphicsZoom_T zoom,
ObjectPool_ScaleFactor_T scaling_type)
```

Parameters

vt_client : VTClient data structure
vt : VT to interact with
callback : Callback when message is sent
object_id : Object ID of the Graphics Context object to modify
viewport_x : Relative x location
viewport_y : Relative y location
zoom : Magnification
scaling_type : Indicates how to scale this object pool part

Returns

bool_t
: TRUE if GraphicsContext_PanAndZoomViewport_Command_Scaled was successful
: FALSE if GraphicsContext_PanAndZoomViewport_Command_Scaled was not successful

GraphicsContext_PanViewport_Command()

F.56 Graphics Context command (Pan Viewport)

Modifies viewport location (Cursor not moved)

This command modifies the viewport X and Y attributes and forces a redraw of the object, allowing "panning" of the underlying object contents. The graphics cursor is not moved.

Signature

```
bool_t GraphicsContext_PanViewport_Command(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id, Pixel_T
viewport_x, Pixel_T viewport_y)
```

Parameters

vt_client : VTClient data structure
vt : VT to interact with
callback : Callback when message is sent

`object_id` : Object ID of the Graphics Context object to modify
`viewport_x` : Relative x location
`viewport_y` : Relative y location

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_SetBackgroundColour_Command()`

F.56 Graphics Context command (Set Background Colour)

Modifies the background colour attribute (cursor not moved)

This command modifies the background colour attribute. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_SetBackgroundColour_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Colour_T colour)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`colour` : New Background Colour

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_SetFillAttributes_Command()`

F.56 Graphics Context command (Set Fill Attributes Object ID)

Modifies the Fill Attribute (cursor not moved)

This command modifies the fill object attribute. All drawing commands that follow use the new attribute value. For no filling, set the object ID to NULL Object ID. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_SetFillAttributes_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
ObjectID_T fill_attributes)
```

Parameters
`vt_client` : VTClient data structure

vt : VT to interact with
callback : Callback when message is sent
object_id : Object ID of the Graphics Context object to modify
fill_attributes : New Fill Attributes

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_SetFontAttributes_Command()

F.56 Graphics Context command (Set Font Attributes Object ID)

Modifies the Font Attribute (cursor not moved)

This command modifies the font object attribute. All drawing commands that follow use the new attribute value. If text is not being used, the object can be set to NULL Object ID. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_SetFontAttributes_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
ObjectID_T font_attributes)
```

Parameters
vt_client : VTClient data structure
vt : VT to interact with
callback : Callback when message is sent
object_id : Object ID of the Graphics Context object to modify
font_attributes : New Font Attributes

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

GraphicsContext_SetForegroundColour_Command()

F.56 Graphics Context command (Set Foreground Colour)

Modifies the foreground colour attribute (cursor not moved)

This command modifies the foreground colour attribute. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_SetForegroundColour_Command(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, Colour_T colour)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`colour` : New Foreground Colour

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_SetGraphicsCursor_Command()`

F.56 Graphics Context command (Set Graphics Cursor)

This command alters the graphics cursor X/Y attributes of the object to the provided absolute position.

Signature
```
bool_t GraphicsContext_SetGraphicsCursor_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
Pixel_T x_position, Pixel_T y_position)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`x_position` : New cursor X position
`y_position` : New cursor Y position

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_SetLineAttributes_Command()`

F.56 Graphics Context command (Set Line Attributes Object ID)

Modifies the Line Attribute (cursor not moved)

This command modifies the Line object attribute. All drawing commands that follow use the new attribute value. For line suppression, set the object ID to NULL Object ID. The graphics cursor is not moved.

Signature
```
bool_t GraphicsContext_SetLineAttributes_Command(const VTClient_T *vt_client,
```

```
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
ObjectID_T line_attributes)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`line_attributes` : New Line Attributes

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_ZoomViewport_Command_Scaled()`

F.56 Graphics Context command (Zoom Viewport) (Scaled)

Modifies viewport magnification (Cursor not moved) Determines the type of scaling to be used when zooming the viewport and calls GraphicsContext_PanAndZoomViewport_Command with the appropriate zoom value.

Signature
```
bool_t GraphicsContext_ZoomViewport_Command_Scaled(const VTClient_T
*vt_client, const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T
object_id, GraphicsZoom_T zoom, ObjectPool_ScaleFactor_T scaling_type)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`zoom` : Magnification
`scaling_type` : Indicates how to scale this object pool part

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`GraphicsContext_ZoomViewport_Command()`

F.56 Graphics Context command (Zoom Viewport)

Modifies viewport magnification (Cursor not moved)

Signature
```
bool_t GraphicsContext_ZoomViewport_Command(const VTClient_T *vt_client,
const VT_T *vt, const ISOBUS_Callback_T *callback, ObjectID_T object_id,
GraphicsZoom_T zoom)
```

Parameters
`vt_client` : VTClient data structure
`vt` : VT to interact with
`callback` : Callback when message is sent
`object_id` : Object ID of the Graphics Context object to modify
`zoom` : Magnification

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`HideShowObject_Command()`

F.2 Hide/Show Object command

Function sends Vt Command(160) HIDE/SHOW OBJECT to the VT

Signature
```
bool_t HideShowObject_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id, ShowHide_Status_T
show_flag)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`object_id` : Object ID
`show_flag` : Object_Hidden or Object_Shown

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`IdentifyVt_Message()`

F.62 Identify VT message

Function sends Vt Command(187) IDENTIFY VT to the VT.

Sends a destination specific IDENTIFY VT command to the VT. If a callback is provided, it will be called when the VT reply is received.

Signature
```
bool_t IdentifyVt_Message(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure

`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`LoadVersion_Callback_Register()`

Registers the load version callback function

Signature

```
void LoadVersion_Callback_Register(VTClient_T *vt_client,
void(*LoadVersion_Response)(VTClient_T *vt_client, const VT_T *vt, const
LoadVersion_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `LoadVersion_Response`

Function pointer to the desired callback function

Returns

(void)

`LoadVersion_Command()`

E.6 Load Version command

This function sends a command to the VT to load the object pool with supplied version label. Note that your working set must have initially sent the object pool that you are requesting to load (i.e., this cannot be used to load another working set's pool), and must have saved it with StoreVersion_Command().

If the version label contains only space characters, then the last object pool stored by your working set will be loaded.

When the VT receives the Load Version command, it will set the "parsing" bit in the VT Status message to 1 until it has finished parsing the object pool. The user can detect this by setting up a callback to receive the Load Version Response message. This is handled with the `LoadVersion_Callback_Register()` function.

Signature

```
bool_t LoadVersion_Command(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, const char *version)
```

Parameters

vt_client
Pointer to the application's VTClient data structure

vt
Pointer to the application's active data structure

callback
Callback when message is sent

version
Version label to load. Must be 7 characters. Any "unused" characters at the end must be padded out with space characters.

Returns

bool_t
`TRUE` Message was sent to the VT `FALSE` Message was not sent (try again later)

`LockUnlockMask_Callback_Register()`

Registers the lock/unlock mask callback function

Signature

```
void LockUnlockMask_Callback_Register(VTClient_T *vt_client,
void(*LockUnlockMask_Response)(VTClient_T *vt_client, const VT_T *vt, const
LockUnlockMask_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `LockUnlockMask_Response`

Function pointer to the desired callback function

Returns

(void)

`LockUnlockMask_Command()`

F.46 Lock/Unlock Mask command

Function sends Vt Command(189) LOCK/UNLOCK MASK to the VT

Sends a destination specific LOCK/UNLOCK MASK command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t LockUnlockMask_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, MaskCommand_T command, ObjectID_T
object_id, Time_T timeout)
```

Parameters

vt_client
Pointer to the application's VTClient data structure

vt
Pointer to the application's active data structure

callback
Callback when message is sent

command
Command: 0 = Unlock Data Mask or Window Mask, 1 = Lock Data Mask or Window Mask

object_id
Object ID of the Data Mask or Window Mask to Lock. If this does not match the visible mask, the command fails with a response code

timeout
Lock timeout in ms or zero for no timeout. Once this period expires, the VT shall automatically release the lock if the Working Set has not done so. This attribute does not apply to an Unlock command

Returns

bool_t
`TRUE` if the message was queued to be sent
`FALSE` if the message was not queued

`PointingEvent_Response()`

H.7 Pointing Event response (optional)

Sends optional response to Pointing Event message

Signature

```
bool_t PointingEvent_Response(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, const PointingEvent_T *message_contents)
```

Parameters

`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns

`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

PointingEvent_Response_Callback_Register()

Registers the pointing event response callback function

Signature

```
void PointingEvent_Response_Callback_Register(VTClient_T *vt_client,
void(*PointingEvent_Message)(VTClient_T *vt_client, const VT_T *vt, const
PointingEvent_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `PointingEvent_Message`

Function pointer to the desired callback function

Returns

(void)

SelectColourMap_Command()

F.60 Select Colour Map command

Function sends Vt Command(186) SELECT COLOUR MAP to the VT.

Sends a destination specific SELECT COLOUR MAP command to the VT. If a callback is provided, it will be called when the VT reply is received.

Signature

```
bool_t SelectColourMap_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id)
```

Parameters

vt_client
VTClient data structure

vt
VT to interact with

callback
Callback when message is sent

object_id
Object ID of the Colour Map object, or NULL_OBJECT_ID to restore the default color table

Returns

bool_t
`TRUE` if the message was queued to be sent
`FALSE` if the message was not queued

SelectInputObject_Command()

F.6 Select Input Object command

Function sends Vt Command(162) SELECT INPUT OBJECT to the VT

Sends a destination specific SELECT INPUT OBJECT command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t SelectInputObject_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, ObjectID_T object_id,
Object_SelectionState_T option)
```

Parameters

vt_client
Pointer to the application's VTClient data structure

vt
Pointer to the application's active data structure

callback
Callback when message is sent

object_id
Object ID - NULL Object ID indicates that no object shall be selected (i.e. focus is removed)

option
Object_NotSelected: Object is not selected
Object_Selected: Set Focus to object referenced by Object ID
Object_SelectedAndOpenForEdit: Activate for data input object referenced by Object ID (invalid for Button Object or Key Object) NOTE Value 0 available only on VT Version 4 and later

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

SetAudioVolume_Callback_Register()

Registers the set audio volume callback function

Signature

```
void SetAudioVolume_Callback_Register(VTClient_T *vt_client,
void(*SetAudioVolume_Response)(VTClient_T *vt_client, const VT_T *vt, const
SetAudioVolume_Response_T *))
```

Parameters

vt_client
VTClient structure containing all active VTs `SetAudioVolume_Response`

Function pointer to the desired callback function

Returns

(void)

`SetAudioVolume_Command()`

F.12 Set Audio Volume command

Function sends Vt Command(164) SET AUDIO VOLUME to the VT

Sends a destination specific SET AUDIO VOLUME command to the VT. If a callback is provided, it will be called when the VT reply is received

Signature

```
bool_t SetAudioVolume_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, AudioVolume_T percent)
```

Parameters

vt_client
VTClient structure containing all active VTs

vt
VT instance

callback
Callback when message is sent

percent
Percent (0 to 100 %) of maximum volume

Returns

bool_t
`TRUE` if the message was queued to be sent
`FALSE` if the message was not queued

`SoftKeyActivation_Response()`

H.3 Soft Key Activation response (optional)

Sends optional response to Soft Key Activation message

Signature

```
bool_t SoftKeyActivation_Response(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, const SoftKeyActivation_T
*message_contents)
```

Parameters

`vt_client`
Pointer to the application's VTClient data structure

`vt`
Pointer to the application's active data structure

`callback`
Callback when message is sent

`message_contents`
Contents of received message

Returns

`bool_t`
`TRUE` if the message was queued to be sent
`FALSE` if the message was not queued

`StoreVersion_Callback_Register()`

Registers the store version callback function

Signature

```
void StoreVersion_Callback_Register(VTClient_T *vt_client,
void(*StoreVersion_Response)(VTClient_T *vt_client, const VT_T *vt, const
StoreVersion_Response_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `StoreVersion_Response`

Function pointer to the desired callback function

Returns

(void)

`StoreVersion_Command()`

Sends the Store Version command to the VT. See LoadVersion_Command() for more details on the `version` parameter.

Signature

```
bool_t StoreVersion_Command(const VTClient_T *vt_client, const VT_T *vt,
const ISOBUS_Callback_T *callback, const char *version)
```

Parameters

vt_client
Pointer to the application's VTClient data structure

vt
Pointer to the application's active data structure

callback
Callback when message is sent

version
Version label

Returns

bool_t
TRUE Message was sent
FALSE Message was not sent (try again later)

VtChangeActiveMask_Response()

H.15 VT Change Active Mask response (optional)

Sends optional response to VT Change Active Mask message

The ECU uses this message to optionally respond to the VT Change Active Mask message.

Signature
```
bool_t VtChangeActiveMask_Response(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, const VtChangeActiveMask_T
*message_contents)
```

Parameters
vt_client : Pointer to the application's VTClient data structure
vt : Pointer to the application's active data structure
callback : Callback when message is sent
message_contents : Contents of received message

Returns
bool_t
: TRUE if the message was queued to be sent
: FALSE if the message was not queued

VtChangeNumericValue_Response()

H.13 VT Change Numeric Value response (optional)

Sends optional response to VT Change Numeric Value message

The ECU uses this message to optionally respond to the VT Change Numeric Value message.

Signature
```
bool_t VtChangeNumericValue_Response(const VTClient_T *vt_client, const VT_T
```

```
*vt, const ISOBUS_Callback_T *callback, const VtChangeNumericValue_T
*message_contents)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`VtControlAudioSignalTermination_Callback_Register()`

Registers the VT control audio signal termination callback function

Signature

```
void VtControlAudioSignalTermination_Callback_Register(VTClient_T *vt_client,
void(*VtControlAudioSignalTermination_Message)(VTClient_T *vt_client, const
VT_T *vt, const VtControlAudioSignalTermination_T *))
```

Parameters

`vt_client`
VTClient structure containing all active VTs `VtControlAudioSignalTermination_Message`

Function pointer to the desired callback function

Returns

(void)

`VtEsc_Response()`

H.11 VT ESC response (optional)

Sends optional response to VT ESC message

The ECU uses this message to optionally respond to the VT ESC message.

Signature
```
bool_t VtEsc_Response(const VTClient_T *vt_client, const VT_T *vt, const
ISOBUS_Callback_T *callback, const VtEsc_T *message_contents)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`VtOnUserLayoutHideShow_Response()`

H.21 VT On User-Layout Hide/Show response (mandatory)

Send VT On User-Layout Hide/Show response

This message applies to Version 4 and later VTs. It is an exception to the other responses specified in this annex in that it is not optional but mandatory: it shall always be sent in response to a VT On User-Layout Hide/Show message.

Signature
```
bool_t VtOnUserLayoutHideShow_Response(const VTClient_T *vt_client, const
VT_T *vt, const ISOBUS_Callback_T *callback, const VtOnUserLayoutHideShow_T
*message_contents)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`VtSelectInputObject_Response()`

H.9 VT Select Input Object response (optional)

Sends optional response to VT Select Input Object message

The ECU uses this message to optionally respond to the VT Select Input Object message

Signature
```
bool_t VtSelectInputObject_Response(const VTClient_T *vt_client, const VT_T
*vt, const ISOBUS_Callback_T *callback, const VtSelectInputObject_T
*message_contents)
```

Parameters
`vt_client` : Pointer to the application's VTClient data structure
`vt` : Pointer to the application's active data structure
`callback` : Callback when message is sent
`message_contents` : Contents of received message

Returns
`bool_t`
: `TRUE` if the message was queued to be sent
: `FALSE` if the message was not queued

`VT_Connect()`

Connects to a VT (start sending Working Set Maintenance)

Signature
```
bool_t VT_Connect(const VTClient_T *vt_client, VT_T *vt)
```

Parameters
`vt_client` : VTClient structure containing all active VTs
`vt` : VT to connect to

Returns
`bool_t`
: `TRUE` Connection started
: `FALSE` Connection not started

`VT_DeleteObjectPool()`

Removes Object Pool from the VT's volatile memory

Signature
```
bool_t VT_DeleteObjectPool(VTClient_T *vt_client, VT_T *vt)
```

Parameters
`vt_client` : VTClient structure containing all active VTs
`vt` : VT to connect to

Returns
`bool_t`
: `TRUE` Disconnection started
: `FALSE` Disconnection not started

`VT_Disconnect()`

Gracefully disconnect from the VT.

Signature

```
bool_t VT_Disconnect(VTClient_T *vt_client, VT_T *vt)
```

Parameters

`vt_client`
VTClient structure containing all active VTs

`vt`
VT to connect to

Returns

bool_t
TRUE Disonnection started
FALSE Disonnection not started

VT_FindVT()

Find active VT structure for given Name Table Index

Signature

```
bool_t VT_FindVT(const VTClient_T *vt_client, const NameTableIndex_T
vt_name_table_index, VT_T **vt)
```

Parameters

Name	Direction	Description
vt_client	In	VTClient structure containing all active VTs
vt_name_table_index	In	Name Table Index of VT
vt	Out	Pointer to VT_T pointer (populated if found)

vt_client
[In] VTClient structure containing all active VTs

vt_name_table_index
[In] Name Table Index of VT

vt
[Out] Pointer to VT_T pointer (populated if found)

Returns

bool_t
TRUE VT found (and **vt populated)
FALSE VT not found

VT_NextVT()

Find the next active VT on the bus

Signature
```
bool_t VT_NextVT(const VTClient_T *vt_client, VT_T **vt)
```

Parameters
vt_client : [In] VTClient structure containing all active VTs
vt : [Out] Pointer to VT_T pointer (populated if found)

Returns
bool_t
: TRUE VT found (and **vt populated)
: FALSE VT not found

VT_SendObjectPool()

Sends an object pool to the VT

Signature
```
bool_t VT_SendObjectPool(const VTClient_T *vt_client, VT_T *vt, const
ObjectPool_T *object_pool)
```

Parameters
`vt_client` : VTClient structure containing all active VTs
`vt` : VT to send object pool to
`object_pool` : Object Pool to send

Returns
`bool_t`
: `TRUE` Connection started
: `FALSE` Connection not started

VTClient_Init()

Initializes the VTClient_T structure

Signature
```
void VTClient_Init(VTClient_T *vt_client)
```

Parameters
`vt_client` : VTClient structure containing all active VTs

Returns

(void)

VTClient_Task()

Runs the VTClient tasks

Signature

```
void VTClient_Task(VTClient_T *vt_client)
```

Parameters

`vt_client`
VTClient structure containing all active VTs

Returns

(void)

VTClient_Uninit()

Uninitializes the VTClient_T structure

Signature

```
void VTClient_Uninit(VTClient_T *vt_client)
```

Parameters

vt_client
VTClient structure containing all active VTs

Returns

(void)

Auxiliary Control API Reference

This section specifies all of the function calls, structures, and macros that make up the VIRTEC Auxiliary Control user interface. For details on any structures, objects, or functions that may be missing here, please see Annexes.h.

Data Types

AuxiliaryAssignmentType2_Error_T: uint8_t
AuxiliaryInputStatusType2_Error_T: uint8_t
AuxInputOperatingState_T: uint8_t
AuxInputValue_T: uint16_t
PreferredAssignment_Error_T: uint8_t

Enumerations

AuxFunctionTypeID_T

Enumeration to indicate the Auxiliary function type

Signature

```
typedef enum AuxFunctionTypeID_E AuxFunctionTypeID_T
```

Members

AuxType_Boolean_Latching
Boolean - Latching (maintains position) On/Off

AuxType_Analog
Analog (maintains position setting)

AuxType_Boolean_Momentary
Boolean - Non-Latching (momentary) Increase value

AuxType_Analog_ReturnToCenter
Analog - return to 50 % Left/Right

AuxType_Analog_ReturnToZero
Analog - return to 0 % Increase value

AuxType_DualBoolean_BothLatching
Dual Boolean - Both Latching (Maintain positions) On/Off/On

AuxType_DualBoolean_BothMomentary
Dual Boolean - Both Non-Latching (Momentary) Increase/Off/Decrease; Raise/Off/Lower

AuxType_DualBoolean_LatchUpMomentaryDown
Dual Boolean - Latching (Up)(Momentary down)

AuxType_DualBoolean_LatchDownMomentaryUp
Dual Boolean - Latching (Down)(Momentary up)

AuxType_Combined_Analog_ReturnToCenter_DualBoolean_Latching
Combined Analog - return to 50% with Dual Boolean - Latching

AuxType_Combined_Analog_DualBoolean_Latching
Combined Analog - maintains position setting with Dual Boolean - Latching

AuxType_QuadratureBoolean_NonLatching
Quadrature Boolean - Non-Latching

AuxType_QuadratureAnalog
Quadrature Analog (maintains position setting)

AuxType_QuadratureAnalog_ReturnToCenter
Quadrature Analog return to 50%

AuxType_BidirectionalEncoder
Bidirectional Encoder

AuxType_RemoveAssignment
Remove Assignment

AuxInputStatus_T

Enumeration for Auxiliary Input Status

Signature

```
typedef enum AuxInputStatus_E AuxInputStatus_T
```

Members

AuxInput_Initializing
Initializing, pool is not currently available for assignment

AuxInput_Ready
Ready, pool has been loaded into the VT and is available for assignments

Structures

AuxiliaryAssignmentType2_T

Signature

```
typedef struct AuxiliaryAssignmentType2_S AuxiliaryAssignmentType2_T
```

Members

ISOBUS_Name_T AuxInputUnit
An index in the ISOBUS NAME table for this Auxiliary Input Unit (device).

bool_t StoreAsPreferred
TRUE to store this assignment as a preferred assignment, FALSE otherwise.

AuxFunctionTypeID_T AuxFunctionType
Auxiliary Function Type. See AuxFunctionTypeID_T in Annexes.h for details.

ObjectID_T AuxInputObjectID
Object ID of Auxiliary Input.

ObjectID_T AuxFunctionObjectID
Object ID of Auxiliary Function.

AuxiliaryFunction_Callback_T

Structure to hold user callbacks.

Signature

```
typedef struct AuxiliaryFunction_Callback_S AuxiliaryFunction_Callback_T
```

Members

void(*AssignmentFunction)(const AuxiliaryAssignmentType2_T *cb_data, bool_t assigned)
AssignmentType2_Command callback.
See (*AssignmentFunction)() for details.

void(*MaintenanceFunction)(const AuxiliaryInputType2Maintenance_Message_T *cb_data)
InputType2Maintenance_Message callback.
See (*MaintenanceFunction)() for details.

void(*StatusFunction)(const AuxiliaryInputType2Status_Message_T *cb_data)
InputType2Status_Message callback.
See (*StatusFunction)() for details.

AuxiliaryFunction_T

Structure to hold auxiliary input on the function side.

Signature

```
typedef struct AuxiliaryFunction_S AuxiliaryFunction_T
```

Members

NameTableIndex_T AuxInputUnit
Auxiliary input unit assigned to this function. (GLOBAL_DEST if not assigned)

NameTableIndex_T PreferredAuxInputUnit
Preferred auxiliary input unit assigned to this function. (GLOBAL_DEST if not assigned)

ObjectID_T AuxInputObjectID
Object ID of auxiliary input assigned to this function (NULL_OBJECT_ID if unassigned)

ObjectID_T AuxFunctionObjectID
Object ID of the auxiliary function

AuxFunctionTypeID_T AuxFunctionType
Auxiliary function type

AuxInputValue_T Value1
Value 1 from auxiliary input

AuxInputValue_T Value2
Value 2 from auxiliary input

AuxInputValue_T PreviousValue1
Previous Value 1 from auxiliary input

AuxInputValue_T PreviousValue2
Previous Value 2 from auxiliary input

SoftwareTimer_T MaintenanceTimer
Timer to track timeout (300 milliseconds) since last Auxiliary Input Type 2 Maintenance message was received

SoftwareTimer_T StatusTimer
Timer to track timeout (300 milliseconds) since last status (momentary) message was received

AuxiliaryFunction_Callback_T *EndUserCallback
Callback function pointer for end user callbacks

AuxiliaryFunctionList_T

Structure containing state information for all active assignments.

Signature

```
typedef struct AuxiliaryFunctionList_S AuxiliaryFunctionList_T
```

Members

Mutex_T Mutex
Mutex containing priority info used for all auxiliary assignments

SoftwareTimer_T WatchdogTimer
Watchdog timer to track task anomalies

AuxiliaryFunction_T *AuxiliaryFunctionArray
Pointer to auxiliary function array

Size_T Size
Size of auxiliary function array

bool_t SendPreferredAssignment
Flag to send the PreferredAssignmentCommand

const PreferredAssignments_Updated_Callback_T *StorePreferredAssignmentCallback
Callback function pointer for user preferred assignment callback

PreferredAssignment_T *PreferredAssignment
Pointer to array of user preferred assignment data

AuxiliaryInput_T

Structure to hold auxiliary input on the input side.

Signature

```
typedef struct AuxiliaryInput_S AuxiliaryInput_T
```

Members

Mutex_T *Mutex
Pointer to mutex containing priority info used for an auxiliary input

ObjectID_T AuxInputObjectID
Object id of the auxiliary input

AuxFunctionTypeID_T AuxFunctionType
Auxiliary function type

AuxInputValue_T Value1
Value 1 from auxiliary input

AuxInputValue_T Value2
Value 2 from auxiliary input

AuxInputValue_T PreviousValue1
Previous Value 1 from auxiliary input

AuxInputValue_T PreviousValue2
Previous Value 2 from auxiliary input

EnableDisable_Status_T Enabled
Boolean to track if auxiliary input is enabled or disabled

SoftwareTimer_T StatusTimer
Timer to track timeout (1 second or 200 milliseconds) since last Auxiliary Input Type 2 Status message was sent

SoftwareTimer_T MinChangeTimer
Timer to track minimum timeout (50 milliseconds) since last Auxiliary Input Type 2 Status message was sent due to input change

AuxiliaryInputList_T

Structure containing state information for all auxiliary inputs.

Signature

```
typedef struct AuxiliaryInputList_S AuxiliaryInputList_T
```

Members

Mutex_T Mutex
Mutex containing priority info used for all auxiliary inputs

AuxInputStatus_T Status
Auxiliary Maintenance Status

SoftwareTimer_T MaintenanceTimer
Timer to track timeout (100 milliseconds) since last Auxiliary Input Type 2 Maintenance message was sent

ModelIdentificationCode_T model_id
Manufacturer defined model identification code

AuxiliaryInput_T *AuxiliaryInputArray
Pointer to auxiliary input array

Size_T Size
Size of auxiliary input array

AuxiliaryInputType2Maintenance_Message_T

Signature

```
typedef struct AuxiliaryInputType2Maintenance_Message_S
AuxiliaryInputType2Maintenance_Message_T
```

Members

NameTableIndex_T AuxInputUnit
An index in the ISOBUS NAME table for this Auxiliary Input Unit (device).

ModelIdentificationCode_T ModelID
Model ID Code of the Auxiliary Input Unit.

AuxInputStatus_T Status
Status. Can be either AuxInput_Initializing or AuxInput_Ready.
AuxInput_Initializing -- pool is not currently available for assignment
AuxInput_Ready -- pool has been loaded onto VT and is ready for assignments

AuxiliaryInputType2Status_Message_T

Signature

```
typedef struct AuxiliaryInputType2Status_Message_S
AuxiliaryInputType2Status_Message_T
```

Members

NameTableIndex_T AuxInputUnit
An index in the ISOBUS NAME table for this Auxiliary Input Unit (device).

ObjectID_T AuxInputObjectID
Aux Input Object ID.

AuxInputValue_T Value1
"Value 1" reported by the Aux Input (see ISO 11783-6, Table J.5)

AuxInputValue_T Value2
"Value 2" reported by the Aux Input (see ISO 11783-6, Table J.5)

AuxInputOperatingState_T OperatingState
Operating State of the Aux Input.
AUX_INPUT_OPERATING_STATE_LEARN_MODE_NOT_ACTIVE (Error code bit 0 == 0)
AUX_INPUT_OPERATING_STATE_LEARN_MODE_ACTIVE (Error code bit 0 == 1)
AUX_INPUT_OPERATING_STATE_INPUT_ACTIVATED_IN_LEARN_MODE (Error code bit 0 == 1, bit 1 == 1)

PreferredAssignment_Response_T

Structure to hold Preferred Assignment Response data.

Signature

```
typedef struct PreferredAssignment_Response_S PreferredAssignment_Response_T
```

Members

PreferredAssignment_Error_T ErrorCodes
Error Codes (0 = no errors)

ObjectID_T AuxFunctionObjectID
Aux Function Object ID of faulty assignment (NULL if no errors)

PreferredAssignments_Updated_Callback_T

Signature

```
typedef struct PreferredAssignments_Updated_Callback_S
PreferredAssignments_Updated_Callback_T
```

Members

void(*StorePreferredAssignmentFunction)()
User PreferredAssignment_T callback

PreferredAssignment_T

Signature

```
typedef struct PreferredAssignment_S PreferredAssignment_T
```

Members

ISOBUS_Name_T Name
indicates the NAME being referenced

ModelIdentificationCode_T model_id
Manufacturer defined model identification code

ObjectID_T AuxInputObjectID
Object ID of auxiliary input assigned to this function (NULL_OBJECT_ID if unassigned)

Macros

MAKE_AuxFunction_Callback_T()

This macro is used to create an element of an AuxiliaryFunction_Callback_T

Signature

```
MAKE_AuxFunction_Callback_T(assign_function, maintenance_function,
status_function)
```

Parameters

assign_function
AssignmentType2_Command end user handler

maintenance_function
InputType2Maintenance_Message end user handler

status_function
InputType2Status_Message end user handler

MAKE_PreferredAssignments_Updated_Callback_T()

Macro used to initialize a PreferredAssignments_Updated_Callback_T structure

Signature
`MAKE_PreferredAssignments_Updated_Callback_T()`

Functions

`(*AssignmentFunction)()`

This function is called when the initial input-to-function assignment is made by the VT, as well as when the assignment is removed (e.g., in case of a status or maintenance message timeout, or if manually removed by the operator -- VIRTEC automatically handles these scenarios).

Signature

```
void(*AssignmentFunction)(const AuxiliaryAssignmentType2_T *cb_data, bool_t assigned)
```

Parameters

`const AuxiliaryAssignmentType2_T *cb_data`
The data from the Aux assignment message.

Note: This data loses scope once the user's callback function returns.
If you need to keep this data for later, make sure to save it into a local buffer.

`bool_t assigned`
`TRUE` the Aux Input is assigned to this Aux Function
`FALSE` the Aux Input is not assigned to this Aux Function

Returns

(void)

`AuxiliaryFunction_Callback_Register()`

Register an end user auxiliary function callback

Signature

```
bool_t AuxiliaryFunction_Callback_Register(const VTClient_T *vt_client, ObjectID_T object_id, const AuxiliaryFunction_Callback_T *callback)
```

Parameters

`const VTClient_T *vt_client`
Applicable VTClient structure

`ObjectID_T object_id`
Object ID of auxiliary function the callback is being registered with. Use `EVERY_OBJECT_ID` to register this callback to every auxiliary function.

`const AuxiliaryFunction_Callback_T *callback`
Callback to be assigned to the auxiliary function

Returns

bool_t
TRUE Callback successfully registered
FALSE Callback registration failed

AuxInput_Analog()

Function for handling an analog input of a given max of 0xFAFF to determine auxiliary input value

Supports the following auxiliary types:

- analog (Aux Function Type ID 1)
- analog return to center (Aux Function Type ID 3)
- analog return to zero (Aux Function Type ID 4)
- combined analog return to center dual boolean latching (Aux Function Type ID 9)
- combined analog dual boolean latching (Aux Function Type ID 10)

Signature

```
void AuxInput_Analog(AuxiliaryInput_T *input, AuxInputValue_T value)
```

Parameters

AuxiliaryInput_T *input
The Aux Input to be updated

AuxInputValue_T value
The current analog value. **Must** be within the range of 0x0-0xFAFF.
Can also be set to ANALOG_LATCHED_BACKWARD or ANALOG_LATCHED_FORWARD to specify that the input is latched.

Returns

(void)

AuxInput_BidirectionalEncoder()

Function for scaling and handling a bidirectional encoder input of a given max with respect to 0xFFFF to determine auxiliary input value

Supports the following auxiliary types:

- bidirectional encoder (Aux Function Type ID 14)

Signature

```
void AuxInput_BidirectionalEncoder(AuxiliaryInput_T *input, AuxInputValue_T
value, AuxInputValue_T rev_counts)
```

Parameters

AuxiliaryInput_T *input
The Aux Input to be updated

AuxInputValue_T value
The current bidirectional encoder value

AuxInputValue_T rev_counts
Number of value counts per revolution

Returns

(void)

AuxInput_Boolean()

Function for handling boolean inputs with 4 bits (left/right/down/up) to determine auxiliary input value.

Supports the following auxiliary types:

- boolean latching (Aux Function Type ID 0)
- boolean momentary (Aux Function Type ID 2)
- dual boolean both latching (Aux Function Type ID 5)
- dual boolean both nonlatching (Aux Function Type ID 6)
- dual boolean latch up momentary down (Aux Function Type ID 7)
- dual boolean latch down momentary up (Aux Function Type ID 8)
- quadrature boolean non latching (Aux Function Type ID 11)

Signature

```
void AuxInput_Boolean(AuxiliaryInput_T *input, AuxInputValue_T left,
AuxInputValue_T right, AuxInputValue_T down, AuxInputValue_T up)
```

Parameters

AuxiliaryInput_T *input
The Aux Input to be updated

AuxInputValue_T left
Current left value
`AUX_ON` == input is pressed left
`AUX_OFF` == the input is not pressed left

AuxInputValue_T right
Current right value
`AUX_ON` == input is pressed right
`AUX_OFF` == the input is not pressed right

AuxInputValue_T down
Current backward value
AUX_ON == input is pressed down/backward
AUX_OFF == the input is not pressed down/backward

AuxInputValue_T up
Current forward value
AUX_ON == input is pressed up/forward
AUX_OFF == the input is not pressed up/forward

Returns

(void)

AuxInput_QuadratureAnalog()

Function for scaling and handling a quadrature analog input of a given max with respect to 0xFAFF to determine auxiliary input value

Supports the following auxiliary types:

- quadrature analog (Aux Function Type ID 12)
- quadrature analog return to center (Aux Function Type ID 13)

Signature

```
void AuxInput_QuadratureAnalog(AuxiliaryInput_T *input, AuxInputValue_T
value1, AuxInputValue_T value2)
```

Parameters

AuxiliaryInput_T *input
The Aux Input to be updated

AuxInputValue_T value1
The current up/down quadrature analog value. **Must** be within the range of 0x0-0xFAFF.

AuxInputValue_T value2
The current left/right quadrature analog value. **Must** be within the range of 0x0-0xFAFF.

Returns

(void)

(*MaintenanceFunction)()

This function is called when the maintenance message is received from an Aux Input assigned to this particular Aux Function.

Signature

```
void(*MaintenanceFunction)(const AuxiliaryInputType2Maintenance_Message_T
*cb_data)
```

Parameters

`const AuxiliaryInputType2Maintenance_Message_T *cb_data`
The data from the Aux Input's maintenance message.

Note: This data loses scope once the user's callback function returns.
If you need to keep this data for later, make sure to save it into a local buffer.

Returns

(void)

`(*StatusFunction)()`

This function is called when the status message is received from an Aux Input assigned to this particular Aux Function.

Signature

```
void(*StatusFunction)(const AuxiliaryInputType2Status_Message_T *cb_data)
```

Parameters

`const AuxiliaryInputType2Status_Message_T *cb_data`
The data from the Aux Input's status message.

Note: This data loses scope once the user's callback function returns.
If you need to keep this data for later, make sure to save it into a local buffer.

Returns

(void)

`VTClient_PreferredAssignments_Get()`

Function for user to save the preferred assignment data to their non-volatile memory.

Signature

```
bool_t VTClient_PreferredAssignments_Get(VTClient_T *vt_client,
Pipe_WriteHandle_T write_handle)
```

Parameters

`VTClient_T *vt_client`
Applicable VTClient to save the PreferredAssignment_T from

`Pipe_WriteHandle_T write_handle`
Pipe to allow user to pipe information to their non-volatile memory

Returns

`bool_t`
`TRUE` Preferred assignment data was successfully saved

FALSE Preferred assignment data failed to save

VTClient_PreferredAssignments_GetSize()

Function for user to get the preferred assignment data size in bytes.

Signature

```
Size_T VTClient_PreferredAssignments_GetSize(const VTClient_T *vt_client)
```

Parameters

const VTClient_T *vt_client
Applicable VTClient the PreferredAssignment_T array is in

Returns

Size_T
Size of the data in the pipe in bytes

VTClient_PreferredAssignments_Set()

Function for user to send the preferred assignment data from their non-volatile memory to the library.

Signature

```
bool_t VTClient_PreferredAssignments_Set(VTClient_T *vt_client,
Pipe_ReadHandle_T read_handle)
```

Parameters

VTClient_T *vt_client
Applicable VTClient to send the PreferredAssignment_T to

Pipe_ReadHandle_T read_handle
Pipe to allow user to pipe information from their non-volatile memory to the library

Returns

bool_t
TRUE Preferred assignment data was successfully sent

FALSE Preferred assignment data failed to send

VTClient_PreferredAssignments_Updated_Callback_Register()

Function for user to register a preferred assignment data callback.

Signature

```
bool_t VTClient_PreferredAssignments_Updated_Callback_Register(const
VTClient_T *vt_client, const PreferredAssignments_Updated_Callback_T
*callback)
```

Parameters

`VTClient_T *vt_client`
Applicable VTClient structure

`const PreferredAssignments_Updated_Callback_T *callback`
Callback (statically located) to be assigned to the auxiliary function working set

Returns

`bool_t`
`TRUE` Preferred assignment data was successfully saved

`FALSE` Preferred assignment data failed to save

Appendix A - Auxiliary Control Global Reference

Aux Function Type 2 Types

Function Type ID	Type	VIRTEC API
0	Boolean - Latching (maintains position) On/Off	AuxInput_Boolean()
1	Analog (maintains position setting)	AuxInput_Analog()
2	Boolean - Non-latching (momentary) Increase value	AuxInput_Boolean()
3	Analog - return to 50% left/right	AuxInput_Analog()
4	Analog - return to 0% Increase value	AuxInput_Analog()
5	Dual Boolean - Both latching (maintain positions) On/Off/On	AuxInput_Boolean()
6	Dual Boolean - Both non-latching (momentary) Increase/Off/Decrease; Raise/Off/Lower	AuxInput_Boolean()
7	Dual Boolean - Latching (up) (momentary down)	AuxInput_Boolean()
8	Dual Boolean - Latching (down) (momentary up)	AuxInput_Boolean()
9	Combined Analog - return to 50% with Dual Boolean - Latching	AuxInput_Analog()
10	Combined Analog - maintains position setting with Dual Boolean - Latching	AuxInput_Analog()
11	Quadrature Boolean - Non-latching	AuxInput_Boolean()
12	Quadrature Analog (maintains position setting)	AuxInput_QuadratureAnalog()

13	Quadrature Analog return to 50%	AuxInput_QuadratureAnalog()
14	Bidirectional Encoder	AuxInput_BidirectionalEncoder()

For more details on the Aux Function Type 2 types, including valid values and ranges, please see ISO 11783-6, Annex J, Table J.5.

Application Notes

Notes on Jetter ISODesigner

Aux Function Type 10 Support
As of version 4.0.6, ISODesigner does not correctly output Aux Function Type 10 objects. After converting the IOP file to C code via ConvertIOP.pl, the user must modify the least significant nibble of Byte 5 (be careful, as this is a bitfield) of the object definition.

From the object pool C file, modifying an Aux Input and an Aux Function object (both Type 10):

```
static const MinAddressable_T App1_SoftKeyMask_array[] =
{
   ...

   /* 28010 - AuxInput2_28010 */
   0x6a, 0x6d, 0x20, 0x01, /* This should be 0x0a, instead of 0x09 ! */ 0x09,
0x01, 0xf1, 0x2e, 0x00, 0x00, 0x00, 0x00,

   ...

   /* 29010 - AuxFunction2_29010 */
   0x52, 0x71, 0x1f, 0x0b, /* This should be 0x8a, instead of 0x89 ! */ 0x89,
0x01, 0x00, 0x2f, 0x00, 0x00, 0x00, 0x00,

   ...
}
```