

This file valid for step 0033.10  
 Issue Date: 2017-03-17

Objects and Models  
 =====

1 Synopsis  
 -----

Models are the "smallest elements" of a Simple Multiuser Scene. The smallest part an author can provide is a model.

Most models will use MIDAS Objects to instrument their functionality (animation, interactivity and simulation).

2 Purpose  
 -----

Models are animated, interactive VR/AR objects that may render real-life objects (RLOs). Models use MIDAS Objects as helpers to calculate quantities for MU capable animation, interactivity and simulation.

\*Intrinsic Models\* cannot be used without their surrounding module (they are actually a part of the module and the module uses the MIDAS Object(s) directly).

\*Bound Models\* are loaded and initialized together with their parent module, but they have been provided independently from the module (bound models can be used by more than one module, more often than once and even by more than one SMS).

\*Unbound Models\* are loaded/unloaded on demand. They can change their attachment to a module during lifetime (handover). Please refer to [052\\_UnboundModels](#) for more information. \*Unbound models have not yet been implemented\*.

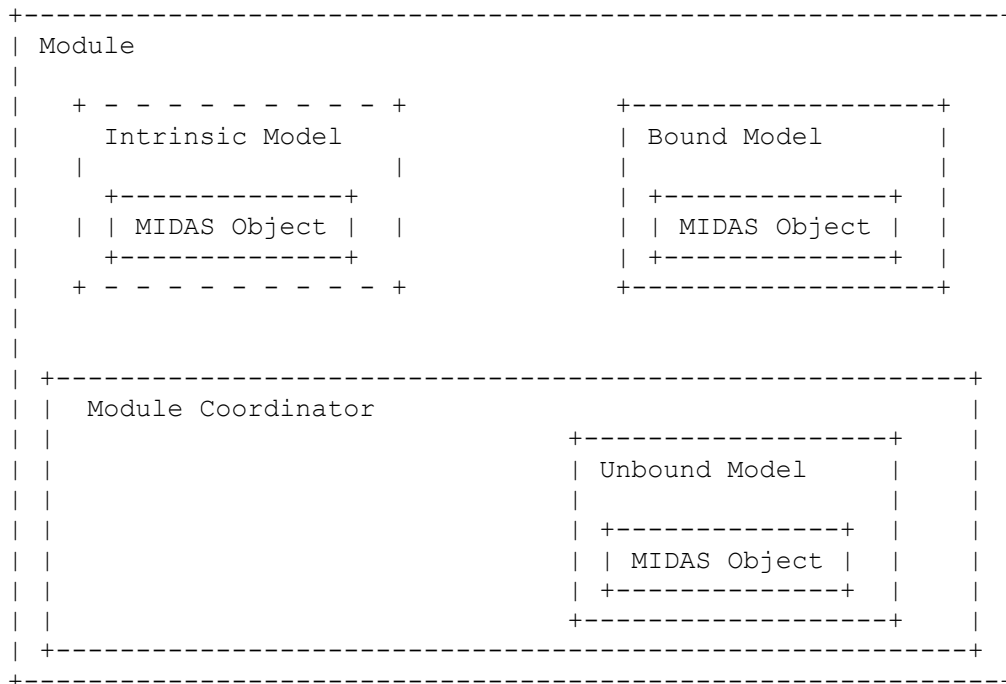


Figure 1: Types of Models and their Context

Models and MIDAS Objects are subsumed under the title "objects". Objects have an object ID and an extended object ID and can hence be uniquely identified within an SMS (see [011\\_NamingRules](#)).

## 2.1 MIDAS Objects

- are loaded and initialized together with their parent model/module\*)
- expose their external interface uiObj either to the containing module (when used in an intrinsic model) or to the containing model\*)
- bound MIDAS Objects are part of an intrinsic model or of a bound model
- unbound MIDAS Objects are part of an unbound model

\*) an exception to these rules are astral objects, which exist outside of any module and outside of any model.  
This is an exceptional case, which is currently only available for the avatar container object to enable the "default avatar container" within the Simple Scene Controller (Base)

## 2.2 Intrinsic Models

- are loaded and initialized as a part of their parent module, and cannot be used without this module
- are positioned relative to the module's coordinate system
- can be interactive and animated, e.g. by using MIDAS objects

## 2.3 Bound Models

- are loaded and initialized together with their parent module (e.g. as <ProtoInstance> node or as <Inline> node)
- are positioned relative to the module's coordinate system
- can be interactive and animated, e.g. by using MIDAS objects
- expose their external interface miModel to their parent module(s)

## 2.4 Unbound Models (\*not yet implemented\*)

- are loaded/unloaded and initialized/disabled by mechanisms of the SRR/SMUOS Framework (Simple Scene Controller, Module Coordinator) and are stored as children of a <Group> node within the Module Coordinator
- contain one or more <Transform> nodes, whose properties are set from the output of the contained "positioning" MIDAS Objects, to position the model
- are displayed relative to the current module's coordinate system
- can be interactive and animated, e.g. by using MIDAS objects
- expose their external interface miModel to the SRR/SMUOS Framework (Simple Scene Controller, Module Coordinator)

Please refer to [052\\_UnboundModels](#) for more information.

### 3 External View

#### 3.1 The External Interface of MIDAS Objects (uiObj)

Each MIDAS Object supports following fields at the external interface uiObj:

- objType
- version
- objId
- parentObj
- commParam
- modParam
- initialized
- attached
- universalObjectClass\*)
- disable\*)
- enabledOut\*)

Additionally, each MIDAS Object supports fields specific to its function.

\*) These fields are supported for use in unbound models (\*not yet implemented\*)

#### 3.2 The External Interface of Models (miModel)

Each model supports at least following fields at the external interface miModel:

- objId
- modParam

Models, that shall be useable as unbound models, support at least following additional fields (\*not yet implemented\*):

- commParam
- initialized
- attached
- universalObjectClass
- disable

Remark: since unbound models need the existence of extensions of the SMUOS Framework, it may happen that some extensions of the SMUOS Framework require more fields from their unbound models than described here.

#### 3.3 Initialization and Attachment - The Modes of Operation (MOOs)

The MIB Core prototype cares about the MOOs. Please see following figure:

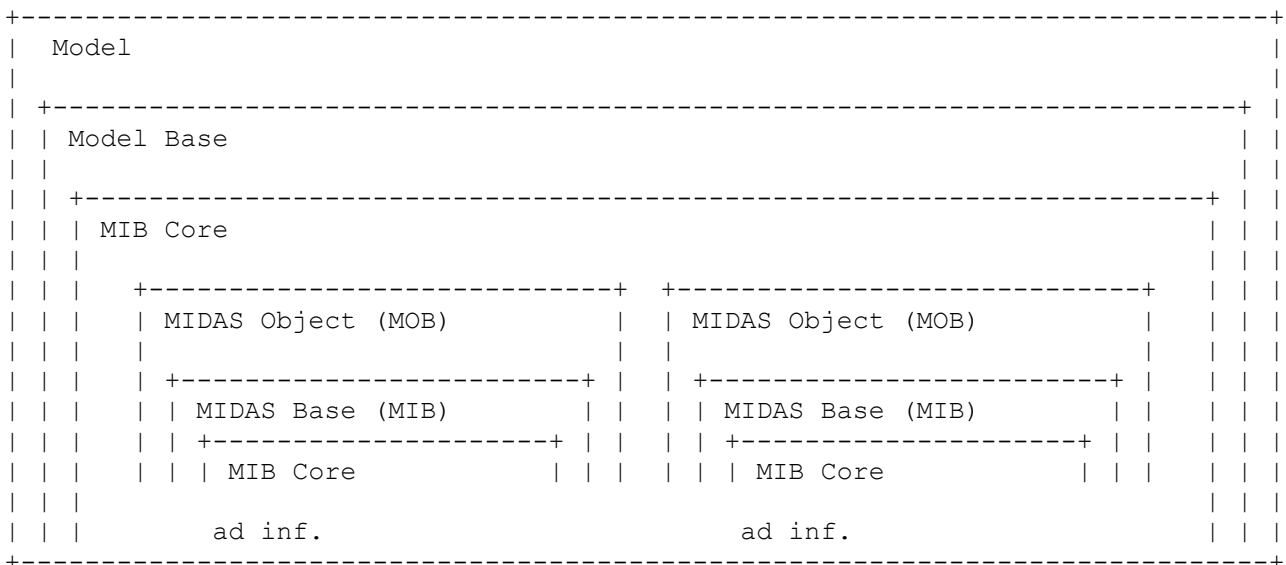


Figure 2: Inner Structure of an SrrTrains/SMUOS Model

That means:

- the MIB Core prototype is used to organize all dependent MIDAS Objects (MOBs)
  - this is valid for a model (which has dependent MOBs), but
  - it is also valid for MOBs, because MOBs can have dependent MOBs, too
- the MIDAS Base prototypes (which should not be confused with the MIB Core prototype) are used within MIDAS Objects only. The MIDAS Base prototypes care about the MAM, the MOC and the ObCo (see next chapter)
- the MIB Core prototype is used to care about initialization and attachment (this chapter)

About the MOOs (Initialization and Attachment):

-----

The field "universalObjectClass" (uoc) can be set only once, immediately after the object has been loaded. This field indicates, whether the object is a bound object or an unbound object

The state "initialized" (MOO I or MOO III) means, the object is ready to function properly, but it is not attached to a module. Being attached to a module would be a precondition to "really" exist, because it's the modules that establish local coordinate systems for being displayed

Being "attached" to a module means the object can have an ObCo (see next chapter), which follows the MOC (see next chapter) of the module. Hence the object can "really" exist.

In some future release the unbound models will have the ability to change their being attached to a parent module (handover)

(Re-)Initialization and an optional De-Attachment is started by sending the "commParam" to the object. The result is

MOO I (initialized) for astral objects (uoc == null)

MOO III (initialized/detached) for unbound objects (uoc != null)

(Re-)Attachment and (Re-)Initialization is started by sending the "modParam" to the object. The result is

MOO II (attached) for bound objects (uoc == null)

MOO IV (attached) for unbound objects (uoc != null)

Disabling is started by sending "disable" to the object or when the parent module of the object gets disabled. Result is

MOO V for astral, bound or unbound objects

Unbound objects can change between MOO III and MOO IV arbitrarily

Astral objects cannot change from MOO I to MOO II (then they were bound)

Bound objects cannot change from MOO II to MOO I (then they were astral)

MOO V cannot be left any more. If an object is once disabled, then it must be unloaded and loaded again to get initialized/attached again

### 3.4 The MAM, the MOC and the ObCo

-----

The module activity matrix (MAM) is already described at [012\\_Modules](#). Each module is inactive in a set of scene instances and active in the complementary set of scene instances. One of the latter scene instances is defined to carry the MOC role for this module.

The Object Controllers (ObCos) of all MIDAS objects follow the MOCs of the parent modules, as long as the MIDAS objects are attached to their parent module.

## 4 Internal View

---

### 4.1 How to Develop MIDAS Objects

---

A short introduction of MIDAS objects can be found in [301\\_MidasObjects](#).

### 4.2 How to Develop Models

---

The demo layout contains a lot of models. Those models can be used as templates for your own models.

### 4.3 Authoring Support

---

The models of the demo layout have been constructed mainly with the help of X3D-Edit, some parts (e.g. the roof of the carousel) were generated with the help of Blender.

It's planned to provide Blender Python Scripts together with the SRR Framework, so that modeling SrrTrains models should become easier in the future.

## 5 Additional Info

---

### 5.1 The Fields of uiObj

---

uiObj is the external interface ("user interface") of a MIDAS Object. Model authors and module authors use the uiObj interface of one or more MIDAS objects to instrument the functionality of their models.

#### 5.1.1 objType, version

---

A MIDAS Object outputs these parameters as some means to identify the object type and the version of the object. This may be helpful in future releases, when MIDAS Objects from different versions ("steps") of the SRR/SMUOS Framework have to work together. Currently some MIDAS Objects use the "objType" field to identify the type of dependent MIDAS Objects (if some type of dependent object is required)

#### 5.1.2 objId

---

Each object needs a unique ID, which distinguishes the object from all other objects of the same module/UOC. The user of the MIDAS Object (module author, model author) has to care, that the object IDs remain unique. When objects are nested, then the nested objects derive their "concatenated object ID" from the "concatenated object ID" of the parent and the own object ID. Please refer to [011\\_NamingRules](#) for further information.

#### 5.1.3 parentObj

---

Before initializing the dependent MOBs, the MIB Core prototype forwards information about the parent object to the dependent MOBs. The presence of this field is presumed by MibCore.

#### 5.1.4 commParam, modParam, disable

---

These fields are forwarded to the contained MIB Core prototype MibCore. These fields are the input for initialization, attachment and disabling of an object. See chapter 3.3. The fields objId and universalObjectClass must be set, BEFORE modParam or commParam is set.

#### 5.1.5 initialized, attached, enabledOut

---

Initialization, ReInitialization, DeInitialization, Attachment, ReAttachment and DeAttachment are triggered by commParam, modParam or disable (see above). The present fields are used to report the MOO of the object after such procedures.

#### 5.1.6 universalObjectClass

---

Some models (unbound models) are not only assigned to a parent (current) module, but also to a so-called "universal object class (UOC)". A SMUOS extension can define one or more UOCs. In this case, the SSC extension has to contain one "SSC Dispatcher" for each UOC. This field takes the reference to the applicable "SSC Dispatcher".

## 5.2 The Fields of miModel

---

miModel is the "minimum interface", that must be provided by a model (minimum set of fields and functionality). This interface is used by module authors to integrate models into their modules and it is used by the SRR/SMUOS Framework to handle unbound models (\*not yet implemented\*).

### 5.2.1 objId

---

Each object needs a unique ID, which distinguishes the object from all other objects of the same module/UOC.

The user of the model (module author, SRR/SMUOS Framework) has to care, that the object IDs remain unique within each module or UOC.

Please refer to [011\\_NamingRules](#) for further information.

### 5.2.2 universalObjectClass

---

Some models (unbound models) are not only assigned to a parent (current) module, but also to a so-called "universal object class (UOC)". A SMUOS extension can define one or more UOCs. In this case, the SSC extension has to contain one "SSC Dispatcher" for each UOC.

This field takes the reference to the applicable "SSC Dispatcher".

### 5.2.3 modParam, commParam, disable

---

These fields are forwarded to the contained MIB Core prototype MibCore. These fields are the input for initialization, attachment and disabling of an object. See chapter 3.3.

The fields objId and universalObjectClass must be set, BEFORE modParam or commParam is set.

### 5.2.4 initialized, attached, enabledOut(?)

---

Initialization, ReInitialization, DeInitialization, Attachment, ReAttachment and DeAttachment are triggered by commParam, modParam or disable (see above).

The present fields are used to report the MOO of the object after such procedures.

### 5.3 General Info about MIDAS Base

---

Each MIDAS Object can be contained in an unbound model or in a bound model or it can be an astral object (MIDAS Objects may be astral, but models can never be astral).

Hence the MIDAS Base (and MibCore) support following types of objects

- astral objects (MOO "LOADED", MOO I or MOO V)
- bound objects (MOO "LOADED", MOO II or MOO V)
- unbound objects (MOO "LOADED", MOO III, MOO IV or MOO V)

Only bound objects and unbound objects can "really" exist, that means they can be attached to a module (MOO II or MOO IV).

When unbound objects are not attached to a module (MOO III aka "detached") then they do not "really" exist, i.e. they cannot have an OBCO in this scene instance and they cannot be active (they cannot deliver quantities for simulation nor animation).

Astral objects are special MIDAS Objects that exist outside of modules and cannot maintain a global state. Models can never be astral objects.

Module independent global state exists either in the SSC or in an unbound model.

In some future release of the SRR/SMUOS Framework it might happen that modules will maintain global state, which is module dependent but somehow "out of the module".



## 5.4 MOO Changes of Objects

---

Objects (i.e. MIDAS Objects and SMS Models) support following MOOs and MOO Changes at their external interfaces uiObj and miModel, respectively:

The MOO Changes are triggered by the fields "commParam", "modParam" and "disable". The actual MOO can be derived from the fields "universalObjectClass", "initialized", "attached" and "enabledOut".

### 5.4.1 Astral Objects

---

Astral Objects support the MOOs

- MOO "LOADED",
- MOO I (initialized) and
- MOO V (disabled)

and following MOO Changes:

- MOO "LOADED" ---> MOO I:
  - Trigger = commParam
  - universalObjectClass = null
  - Procedures:
    - initialization
  - Result:
    - enabled = true (no change)
    - initialized = true (fires)
    - attached = false (no change)
- MOO "LOADED" ---> MOO V:
  - Trigger = disable
  - universalObjectClass = null
  - Procedures:
    - none
  - Result:
    - enabled = false (fires)
    - initialized = false (no change)
    - attached = false (no change)
- MOO I ---> MOO I:
  - Trigger = commParam
  - universalObjectClass = null
  - Procedures:
    - reInitialization
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = false (no change)
- MOO I ---> MOO V:
  - Trigger = disable
  - universalObjectClass = null
  - Procedures:
    - deInitialization
  - Result:
    - enabled = false (fires)
    - initialized = false (fires)
    - attached = false (no change)

## 5.4.2 Bound Objects

-----  
Bound Objects support the MOOs

- MOO "LOADED",
- MOO II (attached) and
- MOO V (disabled)

and following MOO Changes:

- MOO "LOADED" ---> MOO II:
  - Trigger = modParam
  - universalObjectClass = null
  - Procedures:
    - initialization
    - attachment
  - Result:
    - enabled = true (no change)
    - initialized = true (fires)
    - attached = true (fires)
- MOO "LOADED" ---> MOO V:
  - Trigger = disable
  - universalObjectClass = null
  - Procedures:
    - none
  - Result:
    - enabled = false (fires)
    - initialized = false (no change)
    - attached = false (no change)
- MOO II ---> MOO II:
  - Trigger = modParam
  - universalObjectClass = null
  - Procedures:
    - reInitialization
    - reAttachment
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = true (fires again)
- MOO II ---> MOO V:
  - Trigger = disable
  - universalObjectClass = null
  - Procedures:
    - deAttachment
    - deInitialization
  - Result:
    - enabled = false (fires)
    - initialized = false (fires)
    - attached = false (fires)

## 5.4.2 Unbound Objects

-----  
Unbound Objects support the MOOs

- MOO "LOADED",
- MOO III (detached),
- MOO IV (attached) and
- MOO V (disabled)

and following MOO Changes:

- MOO "LOADED" ---> MOO III:
  - Trigger = commParam
  - universalObjectClass != null
  - Procedures:
    - initialization
  - Result:
    - enabled = true (no change)
    - initialized = true (fires)
    - attached = false (no change)
- MOO "LOADED" ---> MOO IV:
  - Trigger = modParam
  - universalObjectClass != null
  - Procedures:
    - initialization
    - attachment
  - Result:
    - enabled = true (no change)
    - initialized = true (fires)
    - attached = true (fires)
- MOO "LOADED" ---> MOO V:
  - Trigger = disable
  - universalObjectClass != null
  - Procedures:
    - none
  - Result:
    - enabled = false (fires)
    - initialized = false (no change)
    - attached = false (no change)
- MOO III ---> MOO III:
  - Trigger = commParam
  - universalObjectClass != null
  - Procedures:
    - reInitialization
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = false (no change)
- MOO III ---> MOO IV:
  - Trigger = modParam
  - universalObjectClass != null
  - Procedures:
    - reInitialization
    - attachment
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = true (fires)

- MOO III ---> MOO V:
  - Trigger = disable
  - universalObjectClass != null
  - Procedures:
    - deInitialization
  - Result:
    - enabled = false (fires)
    - initialized = false (fires)
    - attached = false (no change)
  
- MOO IV ---> MOO IV:
  - Trigger = modParam
  - universalObjectClass != null
  - Procedures:
    - reInitialization
    - reAttachment (may be a "handover")
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = true (fires again)
  
- MOO IV ---> MOO III:
  - Trigger = commParam
  - universalObjectClass != null
  - Procedures:
    - deAttachment
    - reInitialization
  - Result:
    - enabled = true (no change)
    - initialized = true (fires again)
    - attached = false (fires)
    - disabled = false (internal flag)
  
- MOO IV ---> MOO V:
  - Trigger = disable
  - universalObjectClass != null
  - Procedures:
    - deAttachment
    - deInitialization
  - Result:
    - enabled = false (fires)
    - initialized = false (fires)
    - attached = false (fires)