MÄK FOM Editor

Users Guide

VT MÄK
A company of VT Systems

MÄK FOM Editor

Users Guide

VT MÄK
A company of VT Systems

VT MÄK
150 Cambridge Park Drive, 3rd Floor
Cambridge, MA 02140 USA

Voice: 617-876-8085
Fax: 617-876-9208

info@mak.com

www.mak.com

Revision MFE-1.0-1-150420

Contents

Preface

Editing FOMs

Converting HLA 1.3 FOMs to HLA 1516

Index

Preface

This guide is a brief introduction to MÄK FOM Editor to get you started.

How this Manual is Organized

The chapters are organized as follows:

Chapter 1, *Editing FOMs*, explains how to use the FOM Editor to edit your FOM.

Chapter 2, *Converting HLA 1.3 FOMs to HLA 1516*, explains how the FOM Editor converts HLA 1.3 FOMS to support HLA 1516.

MÄK Products

The MÄK FOM Editor is a member of the VT MÄK line of software products designed to streamline the process of developing and using networked simulated environments. The VT MÄK product line includes the following:

- VR-Link® Network Toolkit. VR-Link is an object-oriented library of C++ functions and definitions that implement the High Level Architecture (HLA) and the Distributed Interactive Simulation (DIS) protocol. VR-Link has built-in support for the RPR FOM and allows you to map to other FOMs. This library minimizes the time and effort required to build and maintain new HLA or DIS-compliant applications, and to integrate such compliance into existing applications.

  VR-Link includes a set of sample debugging applications and their source code. The source code serves as an example of how to use the VR-Link Toolkit to write applications. The executables provide valuable debugging services such as generating a predictable stream of HLA or DIS messages, and displaying the contents of messages transmitted on the network.

- MÄK RTI. An RTI (Run-Time Infrastructure) is required to run applications using the High Level Architecture (HLA). The MÄK RTI is optimized for high performance. It has an API, RTIspy®, that allows you to extend the RTI using plug-in modules. It also has a graphical user interface (the RTI Assistant) that helps users with configuration tasks and managing federates and federations.
- VR-Forces®. VR-Forces is a computer generated forces application and toolkit. It provides an application with a GUI, that gives you a 2D and 3D views of a simulated environment.

  You can create and view local entities, aggregate them into hierarchical units, assign tasks, set state parameters, and create plans that have tasks, set statements, and conditional statements. VR-Forces also functions as a plan view display for viewing remote entities taking part in an exercise. Using the toolkit, you can extend the VR-Forces application or create your own application for use with another user interface.
- VR-Vantage™. VR-Vantage is a line of products designed to meet your simulation visualization needs. It includes three end-user applications (VR-Vantage Stealth, VR-Vantage PVD, and VR-Vantage IG) and the VR-Vantage Toolkit.
  - VR-Vantage Stealth displays a realistic, 3D view of your virtual world, a 2D plan view, and an exaggerated reality (XR) view. Together these views provide both situational awareness and the big picture of the simulated world. You can move your viewpoint to any location in the 3D world and can attach it to entities so that it moves as they do.
  - VR-Vantage IG is a configurable desktop image generator (IG) for out the window (OTW) scenes and remote camera views. It has most of the features of the Stealth, but is optimized for its IG function.
  - VR-Vantage PVD provides a 2D plan view display. It gives you the big picture of the simulated world.
  - The VR-Vantage Toolkit is a 3D visual application development toolkit. Use it to customize or extend MÄK's VR-Vantage applications, or to integrate VR-Vantage capabilities into your custom applications. VR-Vantage is built on top of OpenSceneGraph (OSG). The toolkit includes the OSG version used to build VR-Vantage.
- MÄK Data Logger. The Data Logger, also called the Logger, can record HLA and DIS exercises and play them back for after-action review. You can play a recorded file at speeds above or below normal and can quickly jump to areas of interest. The Logger has a GUI and a text interface. The Logger API allows you to extend the Logger using plug-in modules or embed the Logger into your own application. The Logger editing features let you merge, trim, and offset Logger recordings.
- VR-Exchange™. VR-Exchange allows simulations that use incompatible communications protocols to interoperate. For example, within the HLA world, using VR-Exchange, federations using the HLA RPR FOM 1.0 can interoperate with simulations using RPR FOM 2.0, or federations using different RTIs can interoperate. VR-Exchange supports HLA, TENA, and DIS translation.

- VR-TheWorld™ Server. VR-TheWorld Server is a simple, yet powerful, web-based streaming terrain server, developed in conjunction with Pelican Mapping. Delivered with a global base map, you can also easily populate it with your own custom source data through a web-based interface. The server can be deployed on private, classified networks to provide streaming terrain data to a variety of simulation and visualization applications behind your firewall.
- DI-Guy™. The DI-Guy product line is a set of software tools for real-time human visualization, simulation, and artificial intelligence. Every DI-Guy software offering comes with thousands of ready-to-use characters, appearances, and motions. DI-Guy enables the easy creation of crowds and individuals who are terrain aware, autonomous, and react intelligently to ongoing events. Save time, money and create outstanding simulations with DI-Guy. The DI-Guy product line includes the following products:
  - The DI-Guy SDK. Embed the DI-Guy library in your real-time application and populate your world with lifelike human characters.
  - DI-Guy Scenario™. Author and visualize human performances in a rich, user-friendly graphical environment. Use DI-Guy Scenario as an end visualization application or save scenarios and load them into your DI-Guy SDK enabled application.
  - DI-Guy AI. Generate crowds of autonomous characters to quickly populate your worlds with hundreds and thousands of terrain-aware, collision avoiding DI-Guys. Used as a module on top of DI-Guy Scenario and DI-Guy SDK.
  - Expressive Faces Module. Enable DI-Guy characters to have faces that display emotion, eyes that look in directions and blink, and lips that sync to sound files.
  - DI-Guy Motion Editor. Create or customize motions to your particular needs in an easy-to-use graphical application.
- RadarFX. RadarFX is a client-server application that can simulates synthetic-aperture radar (SAR). The server application, which is based on VR-Vantage and SensorFX, loads a terrain database and, optionally, connects to simulations. A client application requests SAR images from the server. MÄK FOM Editor includes a sample client application.

How to Contact Us

For MÄK FOM Editor technical support, information about upgrades, and information about other MÄK products, you can contact us in the following ways:

Telephone

Call or fax us at:	Voice:	617-876-8085 (extension 3 for support)
	Fax:	617-876-9208

E-mail

Sales and upgrade information:	info@mak.com
Technical support:	support@mak.com
VR-Vantage support:	vrv-support@mak.com

Internet

MÄK web site home page:	www.mak.com
License key requests:	www.mak.com/support/get-licenses.html
Product version and platform information:	www.mak.com/support/product-versions.html
For the free, unlicensed MÄK RTI:	www.mak.com/resources/bonus-material/cat_view/16-bonus-materials/24-mak-high-performance-rti.html
MÄK Community Forum:	www.mak.com/community-forum/1-forum.html

Post

Send postal correspondence to:	VT MÄK 150 Cambridge Park Drive, 3rd Floor Cambridge, MA, USA 02140

When requesting support, please tell us the product you are using, the version, and the platform on which you are running.

Document Conventions

This manual uses the following typographic conventions:

`Monospaced`	Indicates commands or values you enter.
`Monospaced Bold`	Indicates a key on the keyboard.
`Monospaced Italic`	Indicates command variables that you replace with appropriate values.
Blue text	A hypertext link to another location in this manual or another manual in the documentation set.
`{ }`	Indicates required arguments.
`[ ]`	Indicates optional arguments.
|	Separates options in a command where only one option may be chosen at a time.
`( | )`	In command syntax, indicates equivalent alternatives for a command-line option, for example, `(-h | --help)`.
/	Indicates a directory. Since MÄK products run on both Linux and Windows PC platforms, we use the / (slash) for generic discussions of pathnames. If you are running on a PC, substitute a \ (backslash) when you type pathnames.
Italic	Indicates a file name, pathname, or a class name.
sans Serif	Indicates a parameter or argument.
➤	Indicates a one-step procedure.
Menu → Option	Indicates a menu choice. For example, an instruction to select the Save option from the File menu appears as: Choose **File → Save**.
	Click the icon to run a tutorial video in the default browser.
	Indicates supplemental or clarifying information.
	Indicates additional information that you must observe to ensure the success of a procedure or other task.

Directory names are preceded with dot and slash characters that show their position with respect to the MÄK FOM Editor home directory. For example, the directory *vrl1.0/doc* appears in the text as *./doc*.

DI-Guy Conventions

Table 1-1 lists the conventions used for entity coordinates in DI-Guy documentation.

Table 1-1: Coordinate system conventions

Convention	Indicates
XYZ	X = forward, Y = to the left Z = up
Yaw, Roll, Pitch	Orientation is applied in the order YRP. Yaw = rz – orientation about the vertical axis Roll = rx – orientation about the fore-aft axis Pitch = ry – orientation nose down, nose up

Mouse Button Naming Conventions

An instruction to click the mouse button, refers to clicking the primary mouse button, usually the left button for right-handed mice and the right button for left-handed mice. The context-sensitive menu, also called a popup menu or right-click menu, refers to the menu displayed when you click the secondary mouse button, usually the right button on right-handed mice and the left button on left-handed mice.

1. Editing FOMs

This chapter explains how to use the FOM Editor.

1.1. Introduction

The MÄK FOM Editor is a tool that allows you to create and edit an HLA Object Model. You can use the tool to create new object models from scratch, or extend and maintain an existing object model from a FOM or FOM Module. Once your changes are made, you can export your object model to HLA 1516-2010 (HLA Evolved) and HLA 1516-2000,. You can export an HLA 1.3 FED file, but not the OMT.

Even if you do not plan to make any changes to your object model, you can still use the tool to view and understand your object model using its intuitive user interface.

To access the MÄK FOM Editor, go to `http://mak.com/Fe/`.

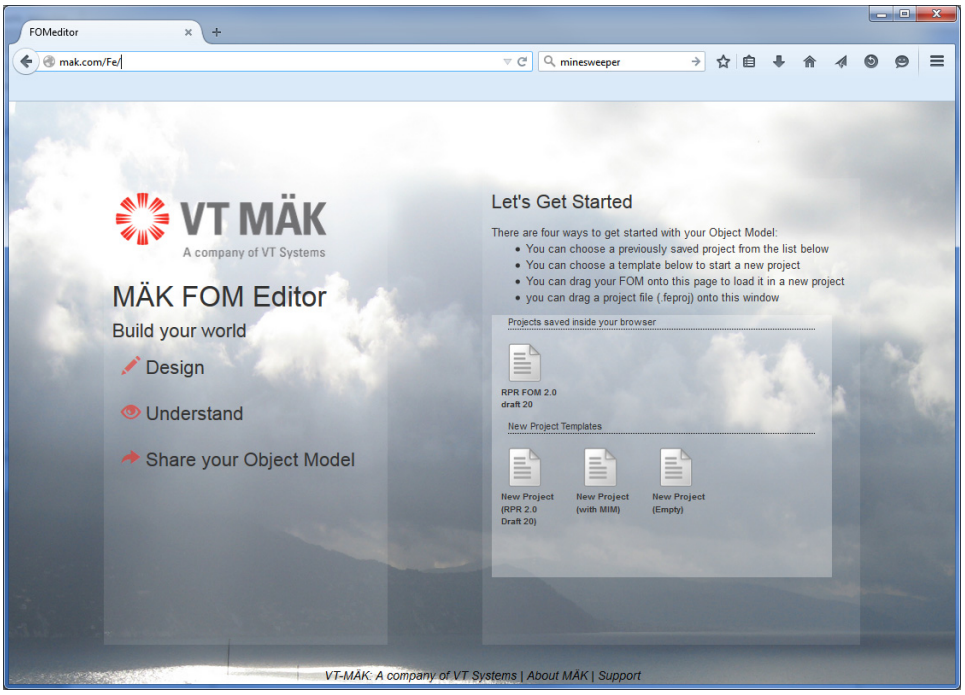

Figure 1-1. MÄK FOM Editor

1.2. Creating a Project

When you use the MÄK FOM Editor, you work in the context of a project. Projects get saved in your browser, not on the MÄK web server or some other internet location.

To create a new project:

1. Do one of the following:
   - Double-click one of the templates listed in the Project panel.
   - Drag a project file (.feproj) onto the Project panel.
   - Drag a FOM file (FDD, or XML) onto the Project panel.

   The MÄK FOM Editor opens the Project page (Figure 1-2).

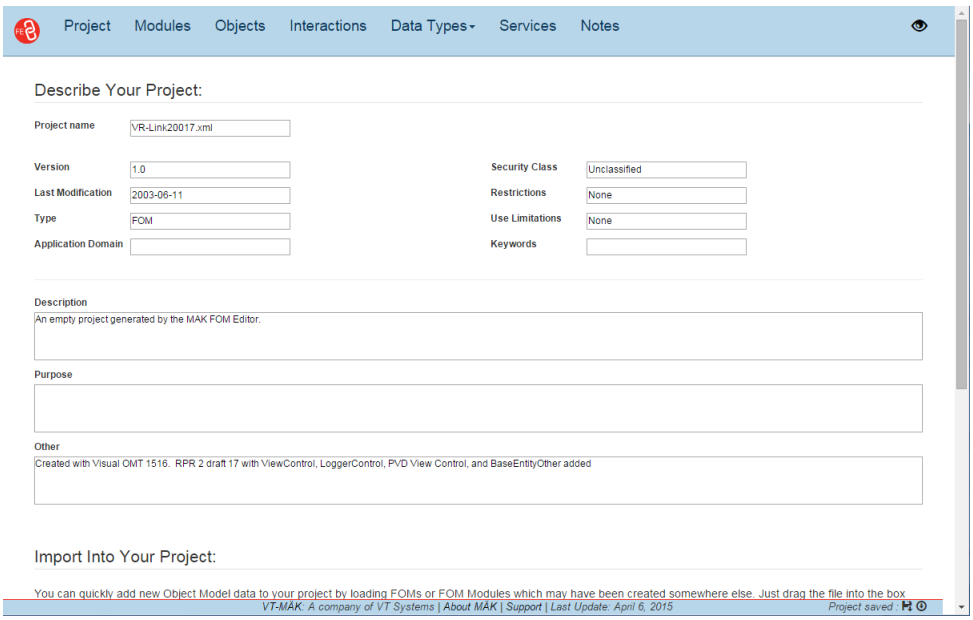

Figure 1-2. Project page

2. If this is a completely new FOM, add a project name, description, and other identifying information. If this project opens an existing FOM, review the descriptive information and update it as needed.
3. Optionally, import data into your project, as described in "Importing Data into a Project," on page 1-6.

1.2.1. Saving a Project

You can save a project in your browser's cache or download it to a FOM Editor project file (*.feproj*).

To save a project to the browser's cache:

1. Click the FOM Editor button. The Project menu is displayed.
2. Choose **Save Project in This Browser**. The project is saved using the Project Name.

Downloading a Project

To download a project:

1. Click the FOM Editor button. The Project menu is displayed.
2. Choose **Download Your Project**. A download dialog box opens.
3. Choose **Open With** or **Save File**.
4. If you choose Open With, select the editor to open the project in. If you choose Save File, the project is saved to the default download location as a file with the extension *.feproj*.

You can also download a project by clicking the Download button ( ) on the Project page.

1.2.2. Clearing the Project Cache

The MÄK FOM Editor stores projects in your browser's cache. You can clear all projects from the cache.

If you clear a project from your cache, you cannot recover it unless you have downloaded it as a project file or have exported the FOM (which you could use to create a new project).

To clear the cache:

1. Click the FOM Editor button. The Project menu is displayed.
2. Choose **Clear Saved Projects**. A confirmation dialog box opens.
3. Confirm or cancel.

1.3. Importing Data into a Project

You can load FOMs and FOM modules into a project.

To import data into a project:

1. Select the Project page.
2. Optionally, specify import options by selecting or clearing the following check boxes:
   - Ignore FOM Module Dependencies.

     HLA 1516-2010 FOM modules may be dependent on one another. Loading a FOM module without first loading the modules on which it depends can lead to problems. Unless this option is selected, the FOM Editor will not allow dependent modules to be loaded until the modules on which they depend are first loaded.
   - Load Project Information From First Module (HLA 1.3 and 1516-2000 only).

     The FOM Editor lets you set project level information that applies to the full FOM, not just individual modules. In HLA 1516-2000 and HLA 1.3, however, the option to use multiple FOM modules is not available. If selected, this option tells the FOM Editor to load project level information from a 1516-2000 FDD file or 1.3 OMT file.
   - Use Standard MOM Classes (HLA 1.3 and 1516-2000 Only).

     HLA 1516-2010, 1516-2000, and 1.3 all define different standard MOM classes. When checked, imported 1516-2000 FDD files and 1.3 OMT files will have MOM classes removed and the internal FOM Editor representation will list 1516-2010 MOM classes. However, on export the 1516-2010 MOM classes will be removed and the appropriate MOM classes will be added to the exported files.
   - Use Lengthless Array Encoding (HLA 1.3 Only).

     In HLA 1.3 the encoding type of arrays was not defined within the OMT. If this check box is cleared, the FOM Editor will define all array data types with the standard HLA 1516 encoding of HLAvariableArray. This encoding consists of a 32 bit integer specifying the number of elements in the array followed by each element of the array. Some FOMs, including the RPR FOM, frequently use a lengthless array encoding, where the length of the array is not explicitly included and the length can instead be inferred from the size of the array and the size of each element. If selected, the FOM Editor will use the RPRlengthlessArray encoding instead of HLAvariableArray.

– Use Null Terminated String Encoding (HLA 1.3 Only).

  In HLA 1.3 a string is defined as a null terminated (0 value) array of ASCII characters. In HLA 1516, the standard string data type is HLAASCIIstring. This data type uses the HLAvariableArray encoding which consists of a 32 bit integer specifying the number of characters followed by an array of ASCII characters. If this option is selected, when a 1.3 OMT file is loaded, the NullTerminatedString data type defined in the RPR FOM, which matches the standard HLA 1.3 string encoding, will be used instead of the HLA 1516 standard HLAASCIIstring data type.

– Use Little Endian Datatypes (HLA 1.3 Only).

  In HLA 1.3 there is no way to specify that a data type should be encoded in big endian or little endian format. If this option is selected, little endian data types will be used. Otherwise, big endian data types will be used.

3. Drag the FOM or FOM module you want to import onto the "Drag your module here" box.

1.4. Exporting a Project

You can export a project to:

- A PDF file.
- A zip file that contains HLA 1516 and HLA Evolved FOM files, a JSON project file, and an HLA 1.3 FED file.
- A MÄK FOM Editor project file. (This is the same thing as downloading a project.)

You can export a project from the Project menu or the Project page.

To export a project from the Project page:

1. On the Project page, select an option under Export Your Project.
2. Click Generate Package.

To export a project from the Project menu:

1. Click the FOM Editor button. The Project menu is displayed.
2. Choose **Export PDF** or **Export FOMs**.

1.5. FOM Modules

HLA Evolved allows you to decompose the object model into logical groups called FOM modules. The use of FOM modules allows federates with different FOMs to interoperate without having to merge the FOM files. The different FOMs must use the same reference FOM, and have non-conflicting extensions. Using modular FOMs supports more agile federations, because only FOM modules pertinent to a particular federate or subset of federates need to be introduced into the federation. Federates can specify one or more FOM modules when creating a federation, and more significantly, a federate can specify one or more FOM modules when it joins. Examples of FOM modules are:

- A normal, fully formed FOM. A federate can create a federation using a FOM just as it always has in the past. It can also submit a FOM when joining, to ensure that the classes it uses are included in the federation.
- A fully formed FOM that is an extension of an existing FOM. It is common to add new classes to a reference FOM to support new simulation concepts or extensions to existing concepts. The use of FOM modules allows these FOM extensions to be introduced by various federates. The extensions do not have to be integrated into a single superset FOM. However, the extensions must not conflict with each other.
- A partial FOM containing new or derivative concepts. New or derivative concepts are supplied in a self-contained FOM module that is not a complete FOM. The new or derivative concepts might depend on concepts found in an existing FOM (especially in the case of derivative concepts).

1.5.1. Adding a FOM Module

When you add a FOM module, you provide descriptive information needed to create it. You add objects and interactions to the FOM module when you edit them on the Objects and Interactions pages. For details, please see "Editing an Object," on page 1-11 and "Editing an Interaction," on page 1-15.

If you are creating a new FOM, you must have at least one FOM module.

To add a FOM module:

1. In an open project, select the Modules page (Figure 1-3).

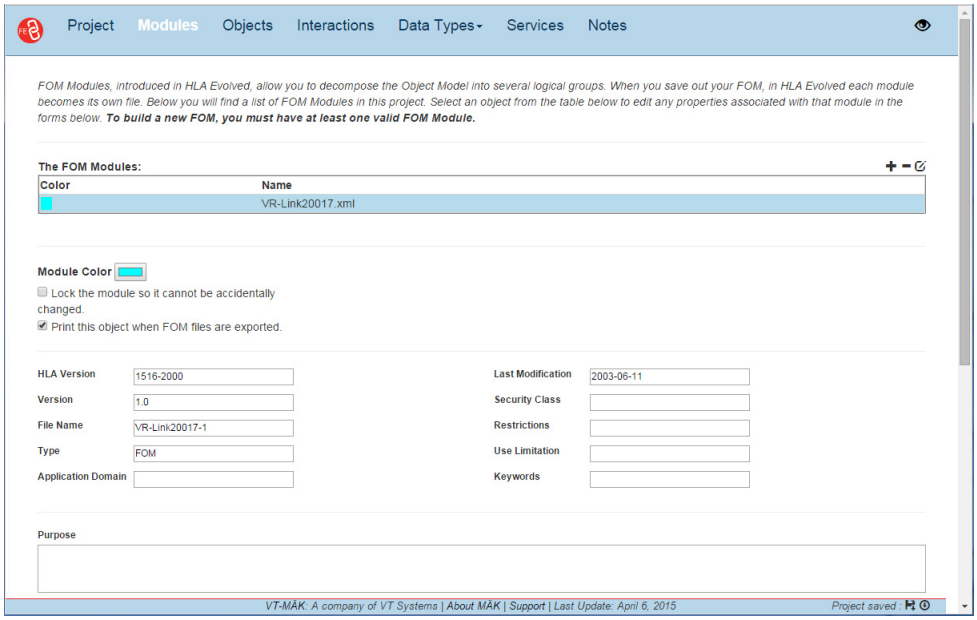

Figure 1-3. Modules page

2. Click the Add button (✚). The Edit FOM Module dialog box opens (Figure 1-4).

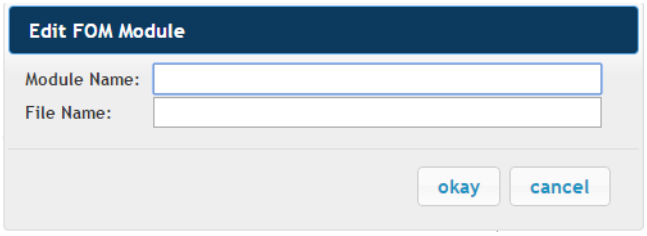

Figure 1-4. Add FOM Module dialog box

3. Specify a name and a filename.
4. Click OK.
5. In the list of FOM Modules, select the module that you added.
6. Optionally, click the color swatch and change the color assigned to the module.
7. Optionally, lock the module so it cannot be edited.
8. Specify whether or not to print the module when the FOM is exported.
9. Optionally, add descriptive information in the various text boxes on the page.
10. Optionally, add dependencies, references, and point of contact information.

Editing FOM Module Dependencies, References, and POCs

A FOM module can be dependent on other modules. You specify the modules that it depends on in the Dependencies table. You can also specify references and points of contact. The procedure for adding, removing, and renaming them is essentially the same.

To add a dependency, reference, or POC:

1. Click the Add button (+) for the table you want to edit. An Edit dialog box opens (for example, Figure 1-5).

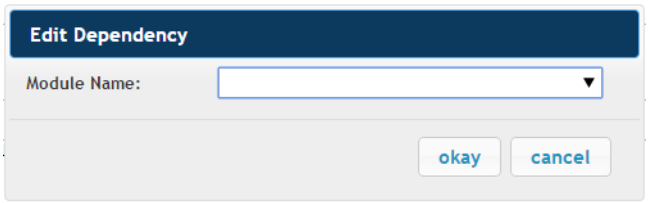

Figure 1-5. Edit Dependency dialog box

2. Type the information required.
3. Click OK. The table is updated.

To edit a dependency, reference, or POC:

1. Select the entry that you want to edit.
2. Click the Edit button (✎). An Edit dialog box opens.
3. Edit the information as desired.
4. Click OK. The table is updated.

To delete a dependency, reference, or POC:

1. Select the entry that you want to delete.
2. Click the Edit button (−). A confirmation prompt opens.
3. Click OK. The entry is deleted.

1.5.2. Deleting a FOM Module

To delete a FOM module:

1. In an open project, select the Modules page.
2. Select the module you want to delete.
3. Click the Delete button (−).
4. Click OK.

1.5.3. Renaming a FOM Module

To rename a FOM module:

1. In an open project, select the Modules page.
2. Select the module you want to rename.
3. Click the Rename button (). The Rename FOM Module dialog box opens.
4. Type a new name.
5. Click OK.

1.6. Objects

The Objects page lists all of the object classes in the FOM and their attributes. You can assign classes to FOM modules.

1.6.1. Editing an Object

To edit an object:

1. Select the Objects page (Figure 1-6).

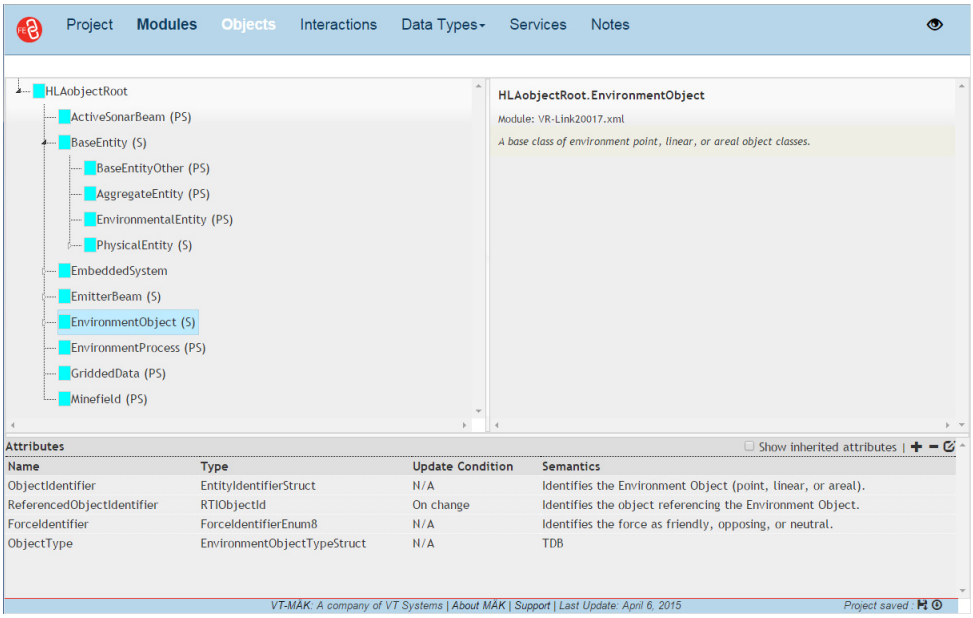

Figure 1-6. Objects page

2. Select the object you want to edit.
3. Right-click the object. A menu is displayed.
4. Select Edit Object. The Object dialog box opens (Figure 1-7).

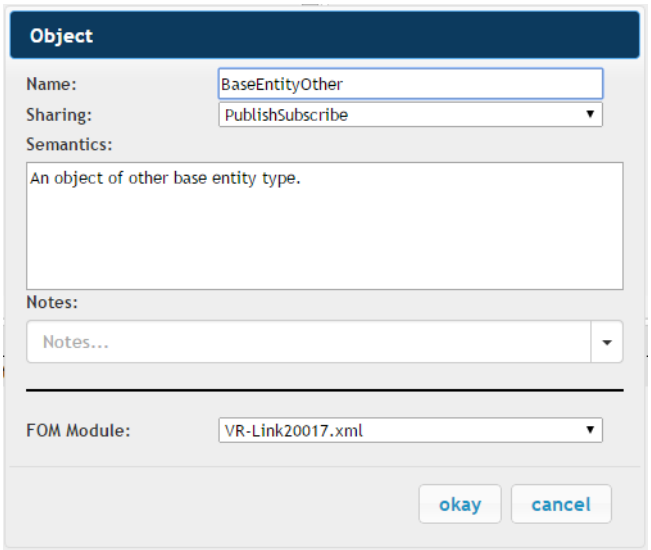

Figure 1-7. Object dialog box

5. Change values as desired.
6. To change the notes associated with an object:
   a. To add a note, in the Notes list, select a note that you want to associate with the object. You can associate more than one note. Just keep selecting additional notes as desired.
   b. To remove a note from the list of associated notes, click the **x** next to the note's number.

1.6.2. Adding a Child Object

To add a child object:

1. Select the Objects page.
2. Select the object you want to add a child to.
3. Right-click the object. A menu is displayed.
4. Select Create a Child Object. The Object dialog box opens (Figure 1-7).
5. Type a name for the child object.
6. Select an option from the Sharing list.
7. Optionally, add a semantic description in the Semantics box.
8. Optionally, add notes.
9. Optionally, specify a FOM module.
10. Click OK.

11. Add attributes, as described in "Adding Object Attributes," on page 1-13

1.6.3. Deleting an Object

To delete an object:

1. Select the Objects page.
2. Select the object you want to delete.
3. Right-click the object. A menu is displayed.
4. Select Delete this Object. A warning prompt opens.
5. Click OK.

1.6.4. Adding Object Attributes

To add an object attribute:

1. In the object hierarchy, select the object for which you want to add an attribute.
2. Click the Add button (✚). The Attribute dialog box opens (Figure 1-8).

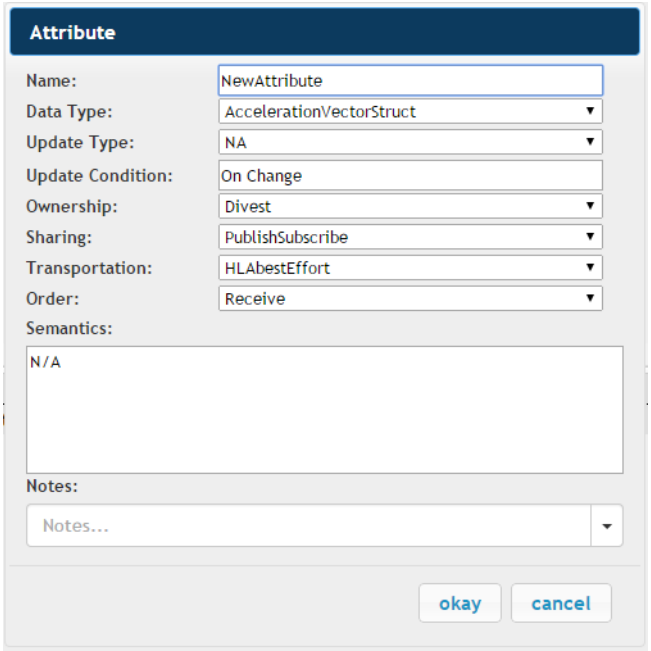

Figure 1-8. Attribute dialog box

3. Type a name for the attribute.
4. Fill in the other attribute values as desired.
5. Click OK.

1.6.5. Editing Object Attributes

To edit an object attribute:

1. In the object hierarchy, select the object for which you want to edit an attribute.
2. In the Attributes list, select the attribute you want to edit.
3. Click the Edit button (). The Attribute dialog box opens (Figure 1-8).
4. Edit values as desired.
5. Click OK.

1.6.6. Displaying Inherited Attributes

By default, the Attributes list just lists the attributes specific to an object. It does not list attributes that the object may have inherited from an ancestor. You can display inherited attributes. You can edit an inherited attribute and the edits apply to the ancestor that owns it.

➤ To display inherited attributes, in the Attributes list title bar, select the Show Inherited Attributes check box.

1.6.7. Deleting Object Attributes

To delete an object attribute:

1. In the object hierarchy, select the object for which you want to delete an attribute.
2. In the Attributes list, select the attribute you want to delete.
3. Click the Edit button (). A warning prompt opens.
4. Click OK.

1.7. Interactions

The Interactions page lists all of the interaction classes in the FOM and their parameters. You can assign classes to FOM modules.

1.7.1. Editing an Interaction

To edit an interaction:

1. Select the Interactions page (Figure 1-9).

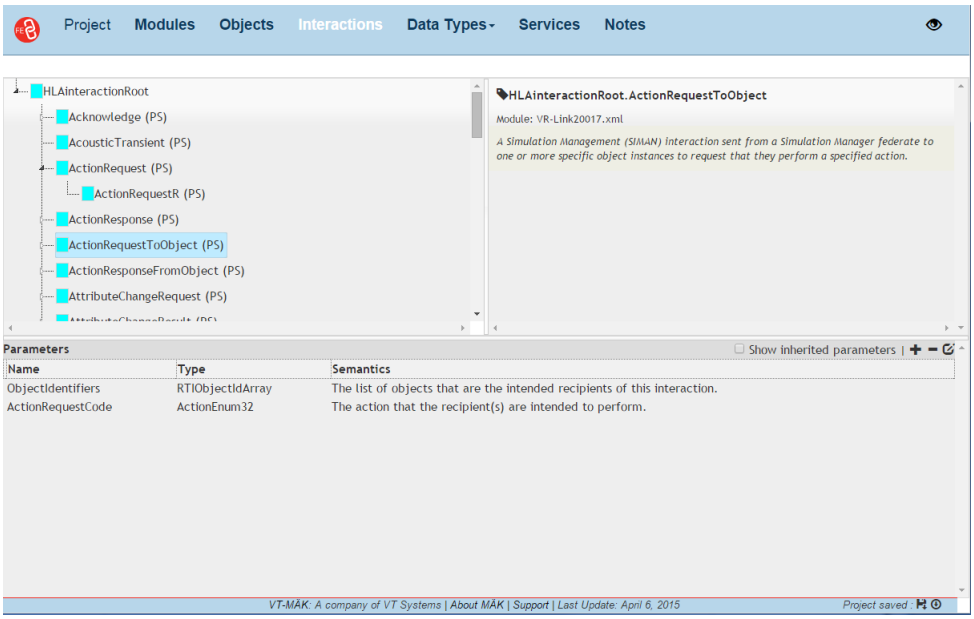

Figure 1-9. Interactions page

2. Select the interaction you want to edit.
3. Right-click the interaction. A menu is displayed.
4. Select Edit Object. The Object dialog box opens (Figure 1-10).

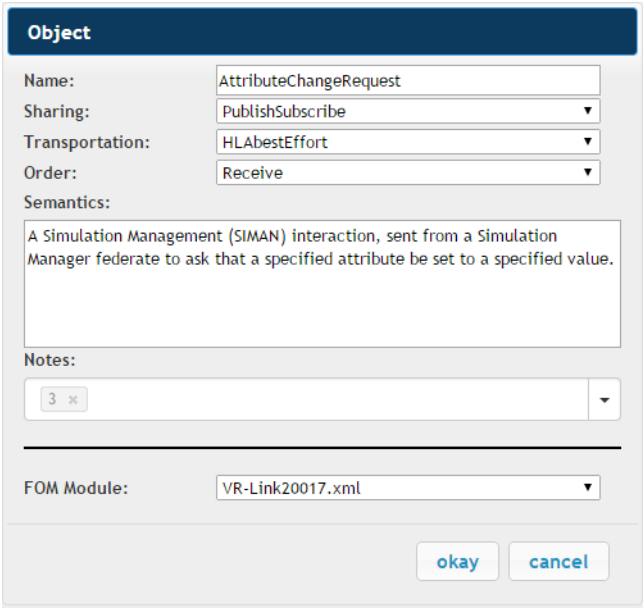

Figure 1-10. Object dialog box for interaction

5. Change values as desired.
6. To change the notes associated with an interaction:
   a. To add a note, in the Notes list, select a note that you want to associate with the interaction. You can associate more than one note. Just keep selecting additional notes as desired.
   b. To remove a note from the list of associated notes, click the **x** next to the note's number.

1.7.2. Adding a Child Interaction

To add a child interaction:

1. Select the Interactions page.
2. Select the interaction you want to add a child to.
3. Right-click the interaction. A menu is displayed.
4. Select Create a Child Object. The Object dialog box opens (Figure 1-10).
5. Type a name for the child interaction.
6. Select an option from the Sharing list.
7. Optionally, add a semantic description in the Semantics box.
8. Optionally, add notes.
9. Optionally, specify a FOM module.

10. Click OK.
11. Add parameters, as described in "Adding Interaction Parameters," on page 1-17

1.7.3. Deleting an Interaction

To delete an interaction:

1. Select the Interactions page.
2. Select the interaction you want to delete.
3. Right-click the object. A menu is displayed.
4. Select Delete this Object. A warning prompt opens.
5. Click OK.

1.7.4. Adding Interaction Parameters

To add an interaction parameter:

1. In the interaction hierarchy, select the interaction for which you want to add an parameter.
2. In the Parameters title bar, click the Add button (✚). The Parameter dialog box opens (Figure 1-11).

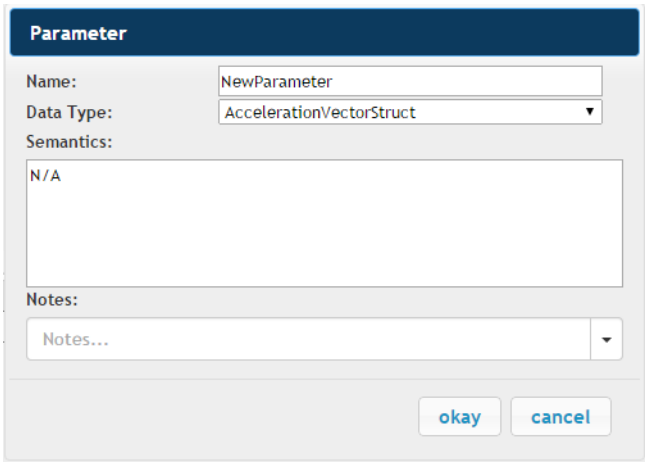

Figure 1-11. Parameter dialog box

3. Type a name for the parameter.
4. Fill in the other parameter values as desired.
5. Click OK.

1.7.5. Editing Interaction Parameters

To edit an interaction parameter:

1. In the interaction hierarchy, select the interaction for which you want to edit a parameter.
2. In the Parameters list, select the parameter you want to edit.
3. In the Parameters title bar, click the Edit button (). The Parameter dialog box opens (Figure 1-11).
4. Edit values as desired.
5. Click OK.

1.7.6. Displaying Inherited Parameters

By default, the Parameters list just lists the parameters specific to an interaction. It does not list parameters that the interaction may have inherited from an ancestor. You can display inherited parameters. You can edit an inherited parameter and the edits apply to the ancestor that owns it.

➤ To display inherited parameters, in the Parameters list title bar, select the Show Inherited Parameters check box.

1.7.7. Deleting Interaction Parameters

To delete an interaction parameter:

1. In the interaction hierarchy, select the interaction for which you want to delete a parameter.
2. In the Parameters list, select the parameter you want to delete.
3. In the Parameters title bar, click the Delete button (). A warning prompt opens.
4. Click OK.

1.8. Data Types

A FOM can have the following data types:

- Basic data.
- Simple data.
- Enumerated data.
- Array data.
- Fixed data.
- Variant data.

Each data type is displayed on its own page.

➤ To view a list of data types, select an option from the Data Types list on the main menu.

1.8.1. HLA 1.3 Data Types

The MÄK FOM Editor can convert HLA 1.3 FOMs to HLA 1516. When you import an HLA 1.3 OMT, the MÄK FOM Editor must make some decisions about converting data types. For details, please see Chapter 2, *Converting HLA 1.3 FOMs to HLA 1516*.

1.8.2. Adding Data Types

To add a data type:

1. Select the page for the data type that you want to add.
2. Click the Add button. A dialog box opens (Figure 1-12).

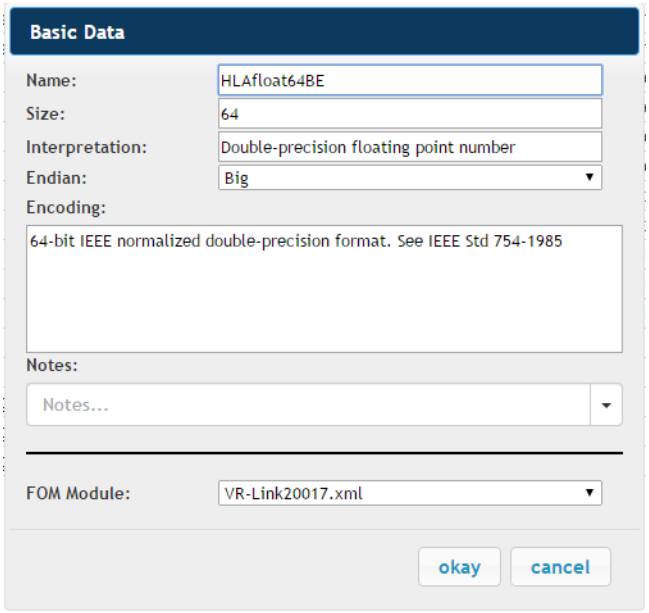

Figure 1-12. Data type dialog box

3. Type a name for the data type.
4. Select a type from the list.
5. Optionally, complete the other data fields for the data type.
6. Click OK.

1.8.3. Editing a Data Type

You can edit the name and characteristics of a data type. You can also make the following edits:

- Enumerated data. Insert, edit, and delete an enumeration.
- Fixed data. Insert, edit, and delete fields and move them up and down.
- Variant data. Insert datum. You can also insert, edit, and delete variant datum.

To edit a data type:

1. Select the page for the data type that you want to edit.
2. Right-click the data type that you want to edit. A menu is displayed.
3. Choose **Edit Data Type**. A dialog box is displayed (Figure 1-12).
4. Edit the data type fields as desired.
5. Click OK.

Editing Enumerated Data Types

For enumerated data types, you can insert, edit, and delete enumerations.

To insert an enumeration:

1. Select the Enumerated Data Types page (Figure 1-13).

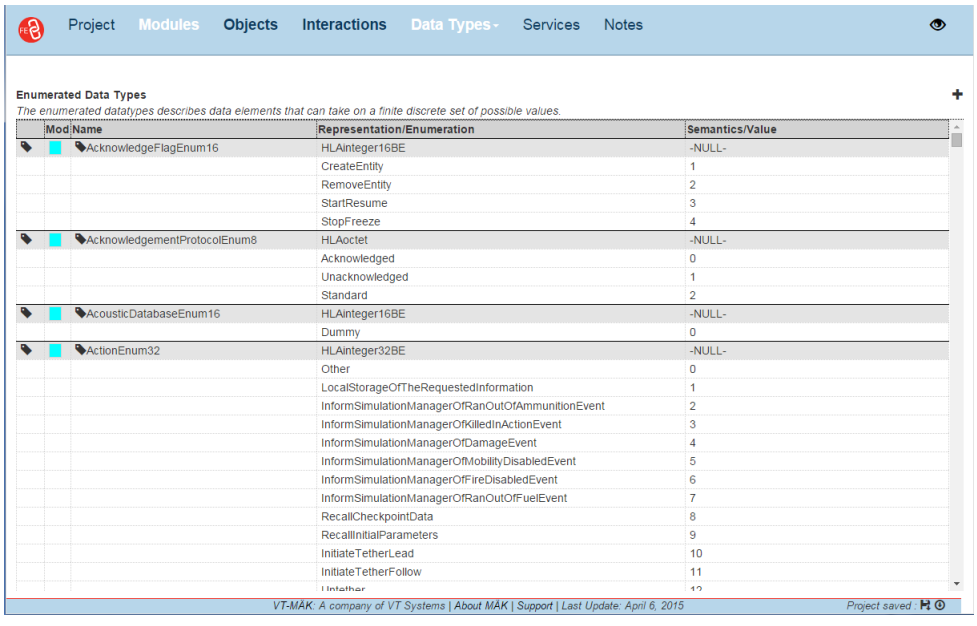

Figure 1-13. Enumerated Data Types page

2. Right-click the enumeration data type that you want to edit. A menu is displayed.

3. Choose **Insert Enumeration**. The Enumerated Data dialog box opens (Figure 1-14).

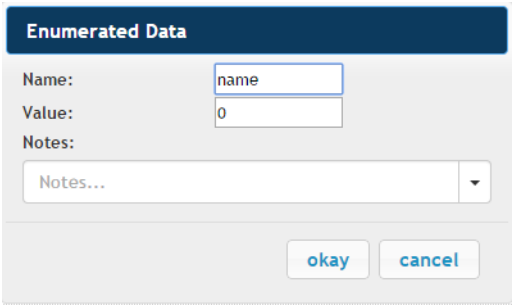

Figure 1-14. Enumerated Data dialog box

4. Specify a name and a value.
5. Optionally specify notes.
6. Click OK.

To edit an enumeration:

1. Select the Enumerated Data Types page.
2. Right-click the enumeration that you want to edit. A menu is displayed.
3. Choose **Edit Enumeration**. The Enumerated Data dialog box opens (Figure 1-14).
4. Edit as desired.
5. Click OK.

To delete an enumeration:

1. Select the Enumerated Data Types page.
2. Right-click the enumeration that you want to delete. A menu is displayed.
3. Choose **Delete Enumeration**. A warning prompt opens.
4. Click OK.

Editing Fixed Data Types

For fixed data types, you can insert, edit, and delete fields.

To insert a field:

1. Select the Fixed Data Types page (Figure 1-15).

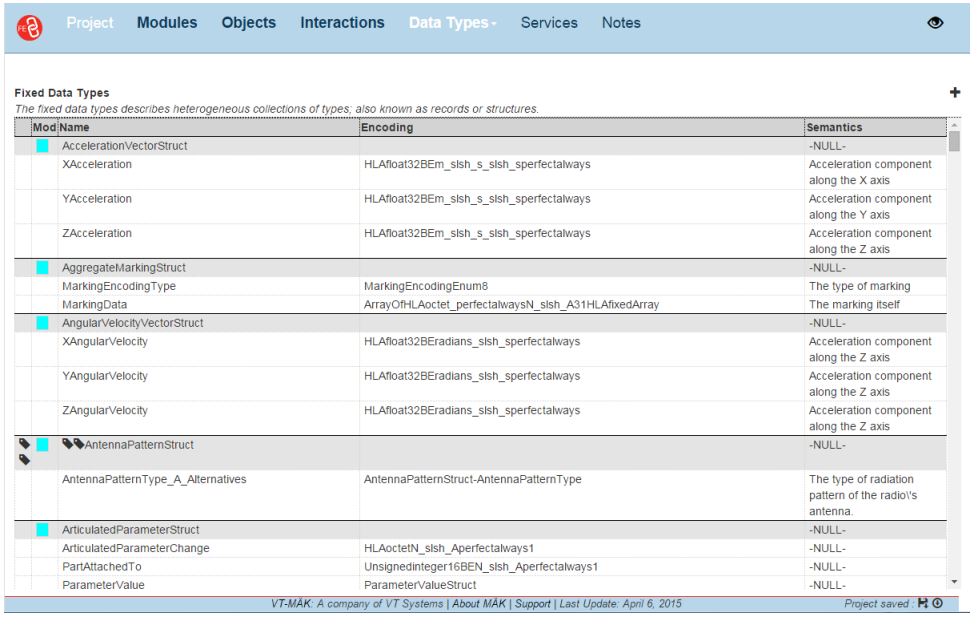

Figure 1-15. Fixed Data Types page

2. Right-click the record that you want to edit. A menu is displayed.
3. Choose **Insert Field**. The Edit Fixed Record Datum dialog box opens (Figure 1-16).

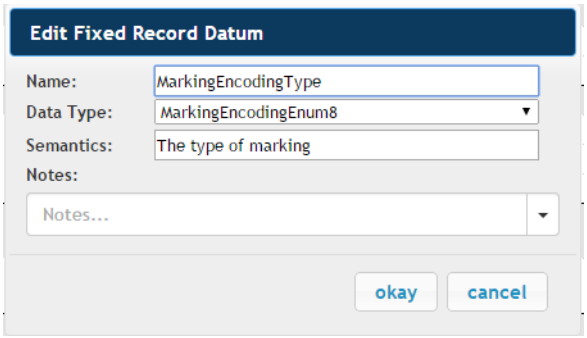

Figure 1-16. Edit Fixed Record Datum dialog box

4. Specify a name and data type.
5. Optionally specify semantics and notes.

6. Click OK.

To edit a field:

1. Select the Fixed Data Types page.
2. Right-click the field that you want to edit. A menu is displayed.
3. Choose **Edit Field**. The Edit Fixed Record Datum dialog box opens (Figure 1-16).
4. Edit as desired.
5. Click OK.

To reorganize fields:

1. Select the Fixed Data Types page.
2. Right-click the field that you want to move up or down. A menu is displayed.
3. Choose **Move Field Up** or **Move Field Down**. The field moves up or down.

To delete a field:

1. Select the Fixed Data Types page.
2. Right-click the field that you want to delete. A menu is displayed.
3. Choose **Delete Field**. A warning prompt opens.
4. Click OK.

Editing Variant Data Types

For variant data types, you can insert, edit, and delete data.

To insert a datum:

1. Select the Variant Data Types page (Figure 1-17).

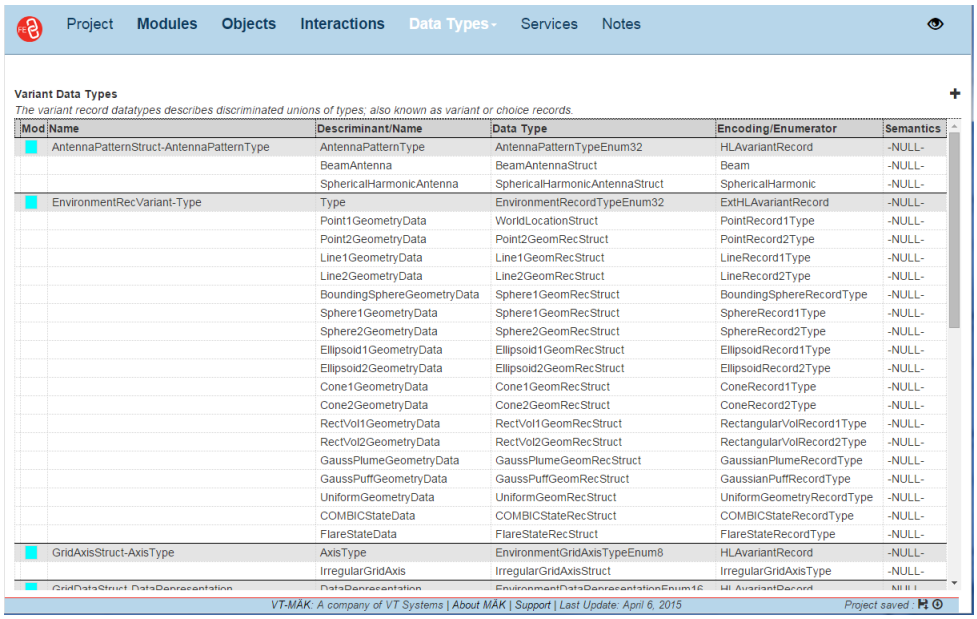

Figure 1-17. Variant Data Types page

2. Right-click the variant data type that you want to edit. A menu is displayed.
3. Choose **Insert Datum**. The Variant Data dialog box opens (Figure 1-18).

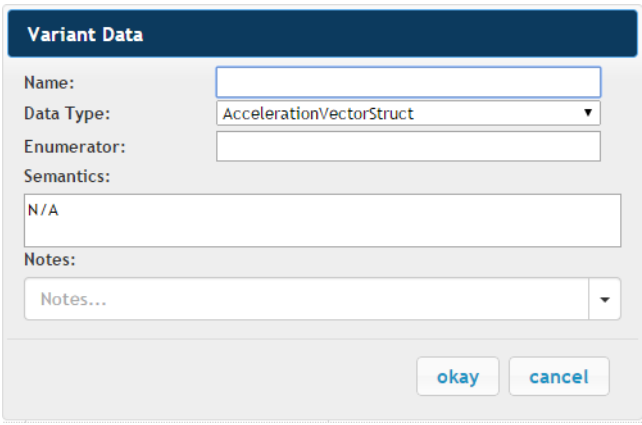

Figure 1-18. Variant Data dialog box

4. Specify a name, a data type, and an enumerator.

5. Optionally specify semantics and notes.
6. Click OK.

To edit a datum:

1. Select the Variant Data Types page.
2. Right-click the variant datum that you want to edit. A menu is displayed.
3. Choose **Edit Variant Datum**. The Variant Data dialog box opens (Figure 1-18).
4. Edit as desired.
5. Click OK.

To delete an enumeration:

1. Select the Variant Data Types page.
2. Right-click the variant datum that you want to delete. A menu is displayed.
3. Choose **Delete Variant Datum**. A warning prompt opens.
4. Click OK.

1.8.4. Deleting a Data Type

To delete a data type:

1. Select the page for the data type that you want to delete.
2. Right-click the data type that you want to delete. A menu is displayed.
3. Choose **Delete Data Type**. A warning prompt opens.
4. Click OK.

1.9. Services

The Services page (Figure 1-19) displays information about:

- Tag types.
- Transportation types.
- Dimensions.
- Synchronization points.
- Update rates.

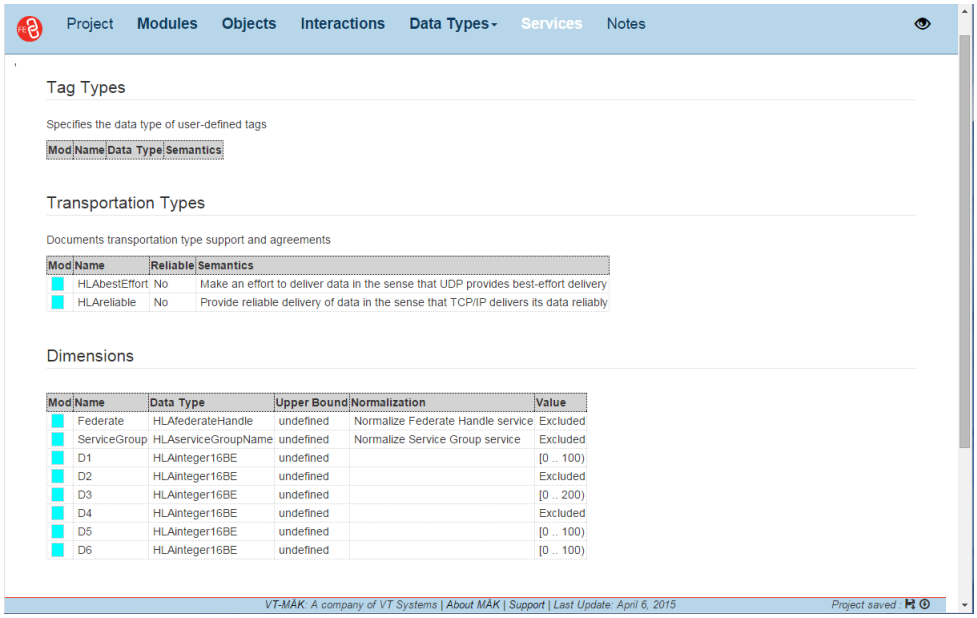

Figure 1-19. Services page

1.10. Notes

A FOM can contain notes that are associated with classes, attributes, parameters, and so on. A note can be associated with more than one thing. A note has a name and text. It can be assigned to a FOM module. In the Notes list, a note's FOM module is identified by its color.

1.10.1. Adding a Note

To add a note:

1. Select the Notes page (Figure 1-20).

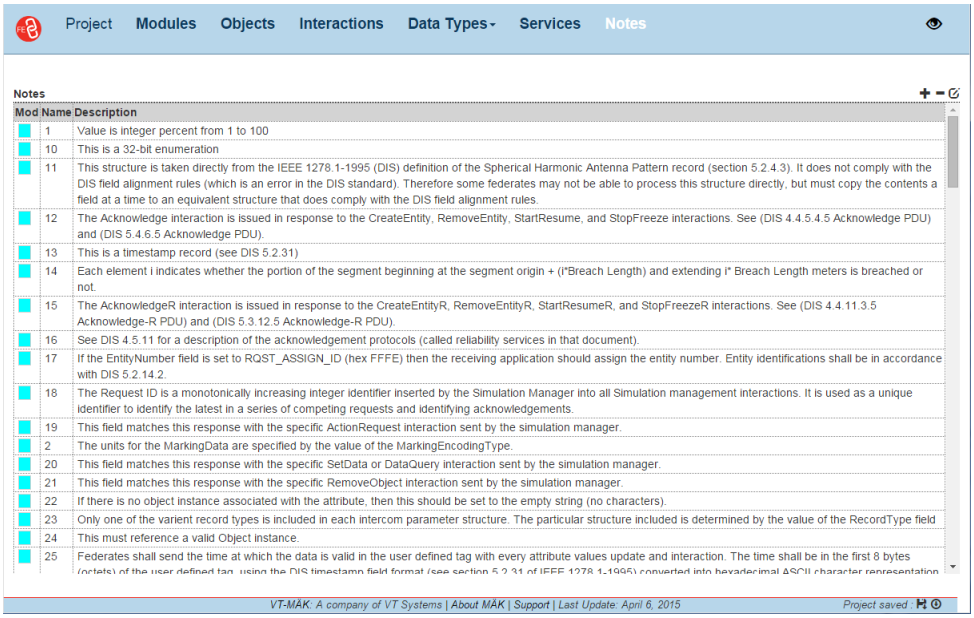

Figure 1-20. Notes page

2. Click the Add button (✚). The Edit Object dialog box opens (Figure 1-21).

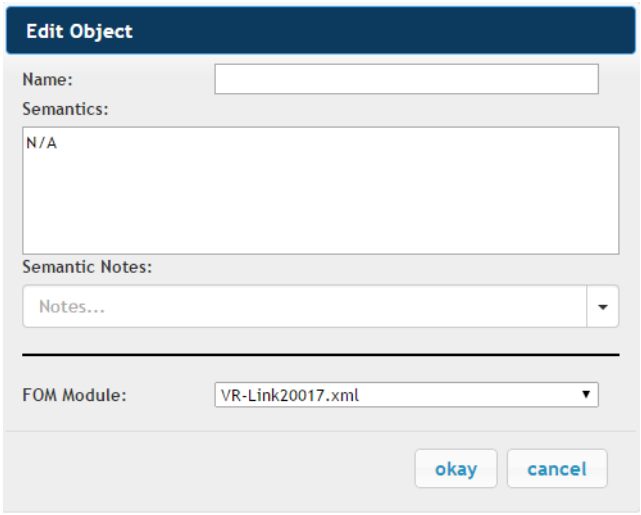

Figure 1-21. Edit Object dialog box

3. Type a name.

4. Type the text of the note in the Semantics box.
5. Optionally, associate other notes with this note.
6. Optionally, select a FOM module from the list.
7. Click OK.

1.10.2. Editing a Note

To edit a note:

1. Select the Notes page.
2. Select the note you want to edit.
3. Click the Edit button (). The Edit Object dialog box opens (Figure 1-21).
4. Edit as desired.
5. Click OK.

1.10.3. Deleting a Note

To delete a note:

1. Select the Notes page.
2. Select the note you want to delete.
3. Click the Delete button (). A warning prompt is displayed.
4. Click OK.

1.11. Inspecting Data

The Data Inspection Panel provides detailed information about the selected data type.

To inspect a data type:

1. Click the Inspection button (). A menu is displayed.
2. Choose **Open Data Panel**. The Data Inspection Panel opens (Figure 1-22).

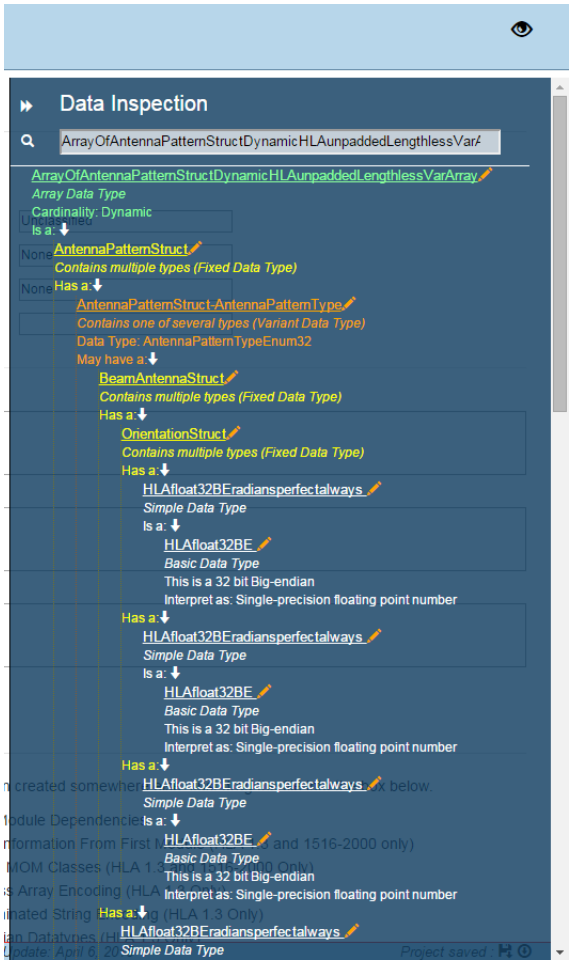

Figure 1-22. Data Inspection panel

3. Select a data type from the list. Details about the data type are displayed. If a line item has an edit icon at the end, clicking the icon opens up an Edit dialog box for the data type.
4. To close the panel, click the right-pointing arrows.

1.12. Validating the Object Model

The FOM Editor can check the object model to make sure that it conforms with the HLA specification.

To validate the object model:

1. Click the Inspection button (). A menu is displayed.
2. Choose **Open Validation Window**. The Object Model Validation dialog box opens (Figure 1-23).

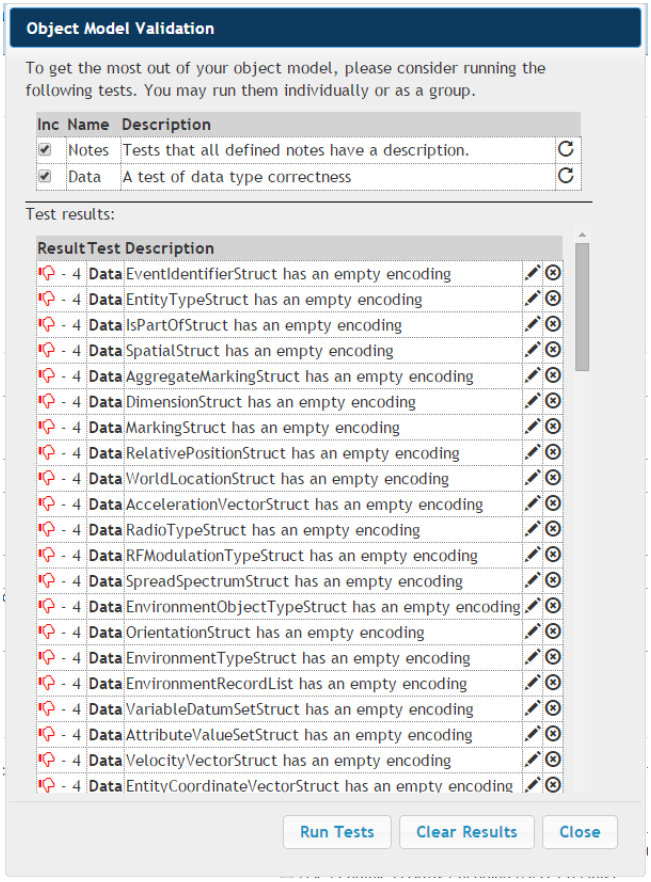

Figure 1-23. Object Model Validation dialog box

3. In the Inc column select the objects you want to validate (notes, data, or both).
4. To run tests on data and notes, click Run Tests. To run tests on just or notes or just data, click the Refresh button for your choice. Results are displayed in the Test results section.

1.12.1. Clearing Test Results

You can delete individual result entries in the Test Results table and you can clear them all at once.

➤ To remove a test result from the list, click the Delete button for that result (⊗).

➤ To clear the test results, click Clear Results.

1.12.2. Editing a Data Type

You can edit a data type from the Test Results list.

To edit a data type:

1. In the Tests Results list, click the edit button (✎) for the data type you want to edit. The Edit Fixed Data dialog box opens .

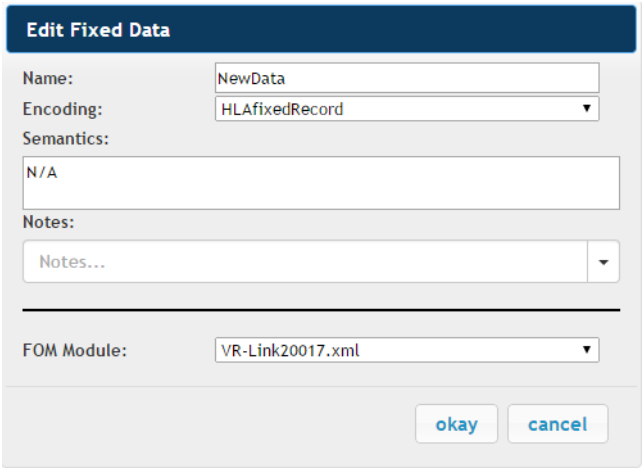

Figure 1-24. Edit Fixed Data dialog box

2. Edit the data as desired.

2. Converting HLA 1.3 FOMs to HLA 1516

This chapter explains how the FOM Editor converts HLA 1.3 FOMs to HLA 1516.

2.1. Introduction

The FOM Editor is designed for creating and editing HLA 1516 FOMs and FOM modules. However it also can convert an HLA 1.3 FOM into an HLA 1516 format. To convert an HLA 1.3 FOM, you must load the OMT file into the FOM Editor on the Project page. The OMT file provides most of the information needed to perform the conversion. The only exceptions are transportation type and order of attributes and interactions. This information is found in the HLA 1.3 FED file. To load these fields as well, you must load the matching FED file after first loading the OMT.

There are a number of differences in the way that HLA 1.3 defines a FOM and the way a FOM is defined in HLA 1516. As a result, the FOM Editor must make some decisions on how best to perform the conversion. The following sections describe how various features of an HLA 1516 FOM are generated from an HLA 1.3 FOM.

2.2. DDM

No DDM information is converted, since DDM configuration is significantly different between 1.3 and 1516.

2.3. Object Classes

Object classes and their attributes are created for each class in the 1.3 OMT file. HLAobjectRoot (and its attribute HLAprivilegeToDeleteObject) is added as the base class.

2.4. Interaction Classes

Interaction classes and their attributes are created for each class in the 1.3 OMT file. HLAinteractionRoot is added as the base class.

2.5. Converting Data Types

HLA 1.3 contains common data representations that are not defined in the OMT file and are expected to be automatically supported. Many of these map to standard HLA 1516 basic data representations. However, some do not exist in HLA 1516. RPR FOM 2.0 has defined additional basic data representations that correspond to these 1.3 data types. Therefore, whenever an appropriate HLA 1516 standard data representation is unavailable, a RPR FOM representation is used where possible. In addition, HLA 1.3 OMT files do not specify the endianness of the data. By default, the FOM Editor will use big endian data types, however an option is available on the Project page to use little endian instead. (For more information, please see "Creating a Project," on page 1-4)

HLA 1.3 to HLA 1516 basic data representations mappings:

- octet: HLAoctet
- float: HLAfloat32BE
- double: HLAfloat64BE
- short: HLAinteger16BE
- unsigned short: RPRunsignedInteger16BE
- long: HLAinteger32BE
- unsigned long: RPRunsignedInteger32BE
- long: HLAinteger64BE
- unsigned long long: RPRunsignedInteger64BE.

A few HLA 1.3 data representations map to HLA 1516 simple, array, or enumerated data types. These are covered in the corresponding sections.

Since 1.3 OMT files do not explicitly define these types, the basic data representations are added to the database as needed. That means that the first time one of these types is referenced in a ComplexComponent, Attribute, or Parameter definition, the corresponding basic data representation will be added to the database.

2.5.1. Simple Data Types

In HLA 1.3, basic data representations (for example, octet, float, long, and so on) are directly referenced in the definitions of ComplexComponents, Attributes, and Parameters. In HLA 1516, however, only simple and enumerated data types directly reference basic data representations. All attributes, parameters, and complex data types must then be constructed using these simple and enumerated types. These simple data types allowed a bit more context to be defined. For instance, a simple data type called DistanceInMeters might be defined that references the basic data representation HLAfloat32BE.

As a result of this difference in data type usage between 1.3 and 1516, the FOM Editor must determine when a new simple data type definition is required. Whenever a basic data representation is used, the FOM Editor will define a new simple data type and give it the name of the ComplexComponent, Attribute, or Parameter that uses it. If an existing data type with that name already exists, the FOM Editor will attempt to determine if it matches the new type. If so, the existing type will be used, and the new type will be dropped. Otherwise, the new type will be renamed with the extension "_1" (or "_2", or "_3", etc.) and added to the database.

The HLA 1.3 basic data representation "char" maps to the HLA 1516 standard simple data type HLAASCIIchar. Therefore, the first time the "char" representation is seen in an OMT file, the simple data type HLAASCIIchar will be added to the database. Since this type depends on the HLA 1516 basic data representation HLAoctet, this representation will also be added.

2.5.2. Array Data Types

In HLA 1.3, array data types are not explicitly defined. Whenever a ComplexComponent, Attribute, or Parameter is intended to define an array, a cardinality specified. This differs from HLA 1516 where all array data types are explicitly defined, including their cardinality. As a result, whenever the FOM Editor encounters a definition in the OMT file that contains a cardinality other than 1, it will define a new array data type and add it to the database. This array data type will be given the name of the ComplexComponent, Attribute, or Parameter that uses it. If an existing data type with that name already exists, the FOM Editor will attempt to determine if it matches the new type. If so, the existing type will be used, and the new type will be dropped. Otherwise, the new type will be renamed and added to the database. If the original name of the new data type did not end with the word "array", the suffix "_Array" will be appended to the name. (In the special case where the array defines a string, the suffix "_String" will be appended instead.) If there is still a conflict with an existing data type a numbered extension such as "_1", "_2", "_3", etc. will be used instead.

If an array data type is composed of basic data representations, a new simple data type will also be added to represent the elements of the array as described in the Simple Datatypes section.

The HLA 1.3 standard does not specify how array data types should be encoded. HLA 1516 array data types include a field that indicates the encoding to use. By default, the FOM Editor will define all array data types with the standard HLA 1516 encoding of HLAvariableArray. This encoding consists of a 32 bit integer specifying the array length followed by each element of the array. Some FOMs, including the RPR FOM, frequently use a lengthless array encoding, where the length of the array is not explicitly included and the length can instead be derived by the size of the array and the size of each element. There is an option on the Project page of the FOM Editor to specify that the RPRlengthlessArray encoding should be used instead.

The HLA 1.3 basic data representation "string" maps to the HLA 1516 standard array data type HLAASCIIstring. Therefore, the first time the "string" representation is seen in an OMT file, the array data type HLAASCIIstring will be added to the database. Since this type depends on the HLA 1516 types HLAASCIIstring and HLAoctet, these will also be added. Some FOMs, including the RPR FOM, frequently use a null terminated string encoding rather than the HLAvariableArray encoding used by HLAASCIIstring. There is an option on the Project page of the FOM Editor to specify that the NullTerminatedString data type should be used in place of HLAASCIIstring.

2.5.3. Fixed Record Data Types

The ComplexDataType definitions found in an HLA 1.3 OMT file convert very naturally into HLA 1516 fixed record data types. A ComplexDataType is composed of multiple ComplexComponents. Each ComplexComponent is converted into a single field in the new fixed record data type. Depending on the data type of the ComplexComponent, this may also lead to the creation of new simple or array data types to represent this field. The Simple Datatypes and Array Datatypes sections have more information on how these types are generated.

If a ComplexDataType contains only a single ComplexComponent, it will be converted to either a simple data type or array data type, depending on the cardinality, instead of a fixed record data type.

2.5.4. Variant Record Data Types

The FOM Editor will never create variant record data types when loading an HLA 1.3 OMT file. While it is possible to represent a variant record using a ComplexDataType, the FOM Editor would require a higher level understanding of the FOM to understand if this is appropriate. Instead, it will create a fixed record data type which can be used to represent a variant record.

2.5.5. Enumerated Data Types

The EnumeratedDataType definitions found in an HLA 1.3 OMT file are converted directly to HLA 1516 enumerated data types. Each of the EnumeratedDataType's Enumeration values is converted to a matching enumerator in HLA 1516. In HLA 1516, each enumerated data type also includes its basic data representation. This information is not present in HLA 1.3 OMT files. As a result the FOM Editor will, by default, use a representation of either HLAoctet, HLAinteger16BE, HLAinteger32BE, or HLAinteger64BE (the equivalent little endian representations will be used if the little endian option is selected on the Project page). The size is determined by the number of bits required to represent the highest value enumerator in the enumerated data type.

The HLA 1.3 basic data representation "boolean" does not map to any standard HLA 156 data type. Instead the FOM Editor will map this representation to the RPR FOM 2.0 standard enumerated data type RPRboolean. The first time the "boolean" representation is seen in an OMT file, the enumerated data type RPRboolean will be added to the database. Since this type depends on the HLA 1516 basic data representation HLAinteger32BE, this representation will also be added.

Index

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V

Link - Simulate - Visualize

MFE-1.0-1-150420

150 CAMBRIDGE PARK DRIVE, 3RD FLOOR CAMBRIDGE, MA 02140 617.876.8085 www.mak.com