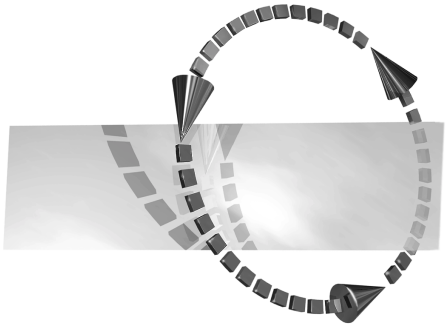

MÄK VR-Link Release 3.4 Release Notes

This document provides the following release-specific information for MÄK VR-Link Release 3.4:

Document ID: VRL-3.4-3-990512

Overview of VR-Link 3.4

VR-Link Release 3.4 offers the following changes and new features:

- Easier license management configuration
- The FOM mapper class and its components are easier to use, and all VR-Link publishers, reflected objects, reflected object lists, and interactions get FOM mapping information from the FOM mapper.
- VR-Link can obtain a FOM mapper from shared libraries that users can create, meaning that you may not need to recompile applications in order to switch FOMs.
- VR-Link supports the entire RPR FOM, versions 0.5, 0.7, and 0.8
- VR-Link works with the DMSO RTI, 1.3v4, 1.3v5, and 1.3v6, and the MÄK RTI 1.3.1.
- The new *./examples/extend directory* contains several fully-documented examples of creating FOM mappers for new FOMs and for extending VR-Link with your own HLA interactions or objects, or DIS PDUs.
- Updated documentation.

System Requirements

Table 1 lists the platforms currently supported by MÄK VR-Link 3.4. Application code must be built with the indicated compilers in order to link to VR-Link libraries.

Table 1: Platforms supported

Platform	Compiler
SGI IRIX6.2 or later (both n32 and o32 ABIs)	MipsPro7.1 C++ compiler (available from SGI)
Sun Sparcstation with SunOS4.1.3 or later	GNU C++ (g++) version 2.7.2.3 and G++ libraries version 2.7.2. (Available via anonymous FTP from prep.ai.mit.edu/pub/gnu.)
Sun Sparcstation with Solaris2.5 or later	SPARCompiler C++ version 4.1 (available from Sun)
PC with Red Hat Linux 5.1	GNU C++ (g++) distributed with Red Hat distribution. The command: `g++ -v` returns: `gcc version egcs-2.90.27 980315 (ecgs-1.02 release)`
PC with Windows NT or Windows 95/98	Microsoft Visual C++ 5.0 or later.

When you use VR-Link and the MÄK RTI on the SunOS4 platform, the link order needs to be a little different from most platforms. Use:

```
-lvlRPR -lRTI -lmatrix -lmtl -lvlutil
```

RTI must precede `mtl` and `vlutil`. The VR-Link example Makefiles reflect this requirement.

Disk Space Requirements

On UNIX, you need approximately 88 MB for VR-Link files and 45 MB for all documentation files.

On PCs, you need 35 MB for VR-Link files and 45 MB for all documentation files.

RTI Support

VR-Link 3.4 has been tested with the MÄK RTI, version 1.3.1, and with the DMSO RTI, versions 1.3v4, 1.3v5, and 1.3v6. Due to incompatibilities between versions of MÄK utility libraries used by VR-Link 3.4 and MÄK RTI 1.3, you will need to move to MÄK RTI version 1.3.1 when you move to VR-Link 3.4.

VR-Link 3.4 comes with MÄK RTI 1.3.1. The installation process on UNIX platforms creates a link called *RTI* in the VR-Link root directory that points to the MÄK RTI 1.3.1 directory.

VR-Link example Makefiles are set up to build against either the DMSO RTI or the MÄK RTI. If you have sourced the DMSO RTI *rti.env* file, VR-Link uses the DMSO RTI, otherwise it uses the MÄK RTI, which it gets to through the *RTI* link in the VR-Link root directory.

When you use the MÄK RTI, remember to make sure that we can find your FED file, and optional *rid.mtl,* file by either putting them in the directory from which you are running, or by setting the environment variable RTI_CONFIG to the directory that contains them.

MÄK RTI 1.3.1 is network compatible with MÄK RTI 1.3, so VR-Link 3.3 applications that use MÄK RTI 1.3 can interoperate with VR-Link 3.4 applications that use MÄK RTI 1.3.1. However, we recommend that you use MÄK RTI 1.3.1 with VR-Link 3.3 applications as well.

RPR FOM Versions Supported

VR-Link 3.4 has built-in support for RPR FOM versions 0.5, 0.7, and 0.8. All RPR FOM classes are supported. The default version is 0.8. We provide FED files for all three versions in the VR-Link root directory. They are called *VR-Link05.fed*, *VR-Link07.fed*, and *VR-Link.fed* (for RPR FOM 0.8).

If you want to use a version other than 0.8, pass the version number (0.5 or 0.7) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExerciseConn* constructor. Also, make sure you are using the right FED file version. For example:

```
DtExerciseConn conn("VR-Link05", "MyApp", new DtRprFomMapper(0.5));
```

VR-Link examples like *f18* and *hlaNetdump* now have a `-f` option that you can use to choose an RPR FOM version by passing either `vlRPR05`, `vlRPR07`, or `vlRPR08` as the argument to `-f`.

At the time of the VR-Link 3.4 release, there were two RPR FOM components that were incorrectly specified in RPR FOM 0.8. They worked in RPR FOM 0.7, but broke during some changes between 0.7 and 0.8. Therefore, we implement these two FOM components as done in RPR FOM 0.7, in our RPR FOM 0.8 FOM Mapper. They are:

- Articulated parts: The structure of the **ArticulatedParametersArray** attribute of the *PhysicalEntity* class is implemented as in RPR FOM 0.7.

- The *RadioTransmitter* class still has an **AntennaPatternType** attribute, as in RPR FOM 0.7. (In RPR FOM 0.8, this is listed as a field of the AntennaPatternStruct complex data type).

You can reduce executable sizes in HLA applications that do not deal with any object classes other than entities and aggregates, or with interaction classes other than fire, detonate and collision, by passing an instance of *DtSimpleRprFomMapper* to the *DtExerciseConn* constructor rather than the default *DtRprFomMapper*.

Backward Compatibility

There is a high degree of backwards compatibility with the VR-Link 3.3 API, particularly if your application does not use the lower-level HLA classes required when extending the toolkit to work with other FOMs. For example, the *f18* example code from VR-Link 3.3 compiles and links unmodified against VR-Link 3.4. If you have derived your own objects or interactions in HLA, some porting will likely be necessary.

Network Compatibility

HLA only

VR-Link 3.4 contains built-in support for RPR FOM versions 0.5, 0.7 and 0.8. VR-Link 3.3 contained built-in support for only RPR FOM version 0.5. If you would like VR-Link-3.4-based HLA applications to communicate with VR-Link-3.3-based HLA applications without doing any special FOM mapping, configure your VR-Link-3.4-based application to use RPR FOM 0.5 by passing:

```
new DtRprFomMapper(0.5)
```

as the mapper argument to the DtExerciseConn constructor.

DIS only

The DIS version of VR-Link still supports the same versions of the DIS protocols, and can therefore interoperate with the older VR-Link versions.

New Features

This section describe the following features and changes to VR-Link:

- FOM mapping
- FOM mapping examples
- Miscellaneous changes to the API
- Non-API changes
- Changes to documentation
- Changes to license management.

FOM Mappin

The way that a *DtExerciseConn* initializes a *DtFomMapper*, and the way that a FOM mapper is configured have changed. See the chapter "HLA-Specific Interface" in the *VR-Link Developer's Guide* for a full description. Here are some highlights:

- You can now pass a *DtFomMapper* to the *DtExerciseConn* constructor, or pass the name of a shared library (DLL or DSO) in which can be found a FOM mapper creation function called DtCreateFomMapper(). This latter option allows you to change an application's FOM mappings without recompiling.
- Within the *DtExerciseConn* constructor, *DtFomMapper's* virtual init() function will be called.
- *DtFomMapper* is an abstract class. There is a new *DtEmptyFomMapper* class that you can instantiate to start with a FOM mapper without any mappings registered.
- *DtFomMapper* and its component classes have many new member functions. For example, instead of setEntityObjectClasses() and setAggregateObjectClasses(), *DtFomMapper* now has setObjectClasses() that takes the name of a *DtReflectedObjectList* subclass, and a set of FOM classes to associate with it.
- *DtHlaObject's* update() function calls a new virtual function called generateUpdate(), which actually generates an outgoing message.

e FOM Mappin and tension amp es

The *./examples/extend* directory contains several examples of configuring VR-Link to work with FOM extensions or changes. See the chapter "FOM Agility Examples" in the *VR-Link Developer's Guide* for details.

Miscellaneous Changes to the API

This section describes changes to the API that are not extensive enough to warrant their own section:

- It is much easier to change the set of classes, attributes and parameters to publish or subscribe to. See the chapter "The HLA-Specific Interface" in the *VR-Link Developer's Guide* for details. As part of this change, *DtObjectClassDesc* and *DtInterClassDesc's* registerInterest() has changed to subscribe().
- There is a simple method to force a publisher to send an update for certain attributes during the next call to tick() even if those attributes' update conditions have not been met. Use DtObjectPublisher::forceUpdate().
- *DtObjectPublisher* has setHlaObject() and detachHlaObject() member functions, so that you can destroy the HLA object being managed by the publisher without destroying the publisher and its state data.
- DtHlaObjectFactory::create() returns a *DtHlaObjectWithStateRep* by default, rather than a *DtUnknownHlaObject*.
- There is an option to have outgoing object updates and interactions reflected back to the sending federate. This option is off by default, but can be enabled by calling DtExerciseConn::setReflecting(DtTrue). This option will only work if there are other federates in the exercise that are subscribed to your data, as otherwise, data will never be sent.
- We have added the classes:
  - *DtGenericStateRepository*
  - *DtGenericObjectPublisher*
  - *DtReflectedGenericObject*
  - *DtReflectedGenericObjectList*.

  These classes allow you to manage HLA objects of classes for which there are no specific state repositories, publishers, and so on. All data is represented as in the RTI, but the job of keeping track of which attributes have been requested, subscribed, and so on, is done for you.
- We have added a *DtSocket* subclass called *DtGroupSocket* that facilitates sending DIS PDUs to multiple point-to-point addresses.
- The *DtString* class is more full-featured.
- The *DtVrlFederateAmbassador* class definition contains "throw" information in function declarations. Not doing so caused problems for some compilers, including C++ 7.2 on IRIX6.
- *DtHashlist* has empty(), keys(), print() and setFromOther() member functions, and a working copy constructor and assignment operator. Also, there is a maximum of one data item per key.
- Dead-reckoning based on absolute timestamps is implemented for HLA (as well as DIS).

- *DtObjectPublisher* and *DtReflectedObject* have localId() member functions.
- *DtHlaObject* has a classDesc() function that returns its class descriptor.
- *DtStateRepository* has localId() and globalId() member functions.
- DtExerciseConn::drainInput() no longer fatally exits when RTI::tick throws an RTI::RTIinternalError exception. The exception now bubbles up to the caller of drainInput().
- DtInteraction::addCallback() can take a single class handle.

on P an es

This section describes other changes to VR-Link:

- When *DtNotifyLevel* is DtNlVerbose, VR-Link prints a diagnostic every time an RTI service is invoked.
- Post-update callbacks registered on *DtReflectedEmitterSystems* are called after updates for constituent beams.
- When computing whether a "heartbeat" PDU needs to be sent, VR-Link compares elapsed real time, not elapsed simulation time to the time threshold.
- DIS applications' executable sizes are much smaller, since now only those PDU class definitions that are needed are pulled in.
- DtStateMsg::print() prints the object instance name in addition to handle.

pdated VR ink Documentation

The *VR-Link Developer's Guide* has been completely revised and re-issued for this release. The Concepts chapter has been significantly expanded as has the Protocol-Independent API. HLA documentation has been brought up-to-date and has extensive new examples of FOM mapping and extending FOMs. The PDU class reference and header file listing, formerly part of the online-only version of the *VR-Link Developer's Guide* are now part of the new *VR-Link Reference Manual*. This manual continues to be available only as a PDF file.

We now provide class documentation based on the header files. It is in PDF format, with extensive hypertext linking to allow you to find information quickly. It includes an alphabetical listing of classes as well as a hierarchical listing. We also provide a printable class hierarchy diagram. All documentation is in the *./do*c directory.

an es to icense Mana ement Setup and on i uration

With this release of VR-Link, we have simplified the process for configuring the FLEXlm license management. It is no longer necessary to put a license file on each client machine, and you no longer have to merge license files when you purchase new products or additional licenses.

If you do not have a prior installation of MÄK Technologies products, follow the instructions for installing license management in the Installation chapter of the *VR-Link Developer's Guide*. If you already have MÄK products installed, your upgrade procedure depends on the version of FLEXlm that you are running. Table 2 lists MÄK products and how to proceed.

Table 2: Product versions and license management

Product	FLEXlm Version	Procedure to Follow
VR-Link 3.0, 3.1 Stealth for SGI 3.1, 3.1.1 Stealth for PCs 3.2.1 Logger 3.1	5.x	Follow installation procedure in original product documentation. Merge license files as described in "Merging New License Files for Products Using FLEXlm 5.x" on page 10.
VR-Link 3.2 or later Stealth for SGI 3.2 or later Stealth for PCs 3.3 or later Logger 3.2 or later Plan View Display 1.x	6.0	Follow the instructions in "Updating Your Existing License Management Configuration" on page 9.

pdatin our istin icense Mana ement on i uration

If you have FLEXlm license management configured for products you purchased prior to the current release, you must do the following to update your license management configuration:

1. On the license server machine, rename the *license.dat* file to *mak.lic* (or any name ending in *.lic*). (If you are unsure about the distinction between the license server machine and client machines, see "The FLEXlm Client-Server Architecture" in the Installation chapter of VR-Link Developer's Guide.)
2. Open the license file in a text editor.
3. Remove the port number from the SERVER line. For example, if the first line of the license file is:

   ```
   SERVER oak 0800359de785 4567
   ```

   the port number is 4567. Delete it from the file.
4. Remove the LM_LICENSE_FILE environment variable from the server machine and all client machines unless another vendor requires it.

5. Delete the *license.dat* file from all client machines.
6. Set the MAKLMGRD_LICENSE_FILE environment variable on all client machines (including the license server machine if you run MÄK Technologies products on it).

   The syntax for the environment variable is: *@Server_name*. For example, if the server machine is oak, set the environment variable to `@oak`. (If you are not sure how to set environment variables on your machine, see the section "Setting the MAKLMGRD_LICENSE_FILE Environment Variable" in the Installation chapter of *VR-Link Developer's Guide*.)

Mer in e icense Fi es or Products sin F m

If you are using FLEXlm 5.x, you cannot use the new method of license management. If you receive a new license file from MÄK, you need to merge it with the old file.

1. Your new license file will look something like the following:

```
SERVER mollusc 006097752f93
DAEMON maklmgrd .
PACKAGE dir maklmgrd 1999.177 60D00041FA200D8A795D \
      COMPONENTS=dir1
INCREMENT dir maklmgrd 1999.177 1-jan-0 1 0BB480511615C175B8CF \
      PLATFORMS=i86_n
```

   Your existing *license.dat* file is similar to the following:

```
SERVER mollusc 006097752f93 4567
DAEMON maklmgrd "c:\Program Files\MÄK Technologies\stealth\flexlm\
      maklmgrd"
PACKAGE makproduct maklmgrd 1999.177 40C020D12D609AB04D55 \
      COMPONENTS="hprod1 dprod1 prod1"
INCREMENT makproduct maklmgrd 1999.177 1-jan-0 1 AB4450916DD1944435D2 \
      PLATFORMS=i86_n
```

Notes

- The server name and host id in the two license files should be identical. The new file will not have a port number.
- The line continuation marks (\) in the examples are for clarity of presentation. They may not be present in the files you receive.

2. Open up the two files in a text editor. Append everything except the Server and Daemon lines (the Server lines will be identical except for the port) from the new *license.dat* file to the end of the old *license.dat* file. For example:

```
SERVER mollusc 006097752f93 4567
DAEMON maklmgrd "c:\Program Files\MÄK Technologies\stealth\flexlm\
      maklmgrd"
PACKAGE makproduct maklmgrd 1999.177 40C020D12D609AB04D55 \
      COMPONENTS="hprod1 dprod1 prod1"
INCREMENT makproduct maklmgrd 1999.177 1-jan-0 1 AB4450916DD1944435D2 \
      PLATFORMS=i86_n
PACKAGE dir maklmgrd 1999.177 60D00041FA200D8A795D \
      COMPONENTS=dir1
```

```
INCREMENT dir maklmgrd 1999.177 1-jan-0 1 0BB480511615C175B8CF \
      PLATFORMS=i86_n
```

3. Put the updated *license.dat* file in the *flexlm* directory on the license server and on all machines on which you have VR-Link installed.

Bug Fixes

The following problems in VR-Link 3.3 have been fixed:

- In PDU classes derived from *DtPduWithData*, if you changed the size of a datum param it wouldn't always take effect. This problem depended on the previous value for that particular length field. If the new length was in the same block of 8 bytes as the previous length, the previous length would remain as the size, not the desired new length.
- In the class *DtDataQueryPdu*, the timeInterval() function returned erroneous results.
- In VR-Link 3.3, if you subscribed to an object class before creating a *DtReflected-ObjectList* that managed it, the *DtReflectedObjectList* would not be made aware of objects for which we received RTI discoverObjectInstance calls before the *DtReflectedObjectList* was created.
- Fixed a bug in decoding the EventID and Marking attributes of several classes. Sometimes, garbage characters would get tacked onto the end of these fields in *DtEntityStateRepository.*
- Our encoder and decoder factories' addAttributeDecoder/Encoder and addParameterDecoder/Encoder functions did not work correctly in VR-Link 3.3. This has been fixed.
- We no longer crash if we get a provideAttributeUpdate service invocation for objects that we don't know about. This will occur if you register some of your objects directly with the RTI, rather than through VR-Link.
- The TacticalDataLink parameter of the *RadioSignal* interaction was incorrectly interpreted as 16 bits rather than 32.
- Fixed DtString::operator+().
- Fixed several memory leaks.

Known Problems

Applications that can load their own *.mtl* files (like our *f18* example) can hang when exiting, on IRIX6o32 and Linux platforms.