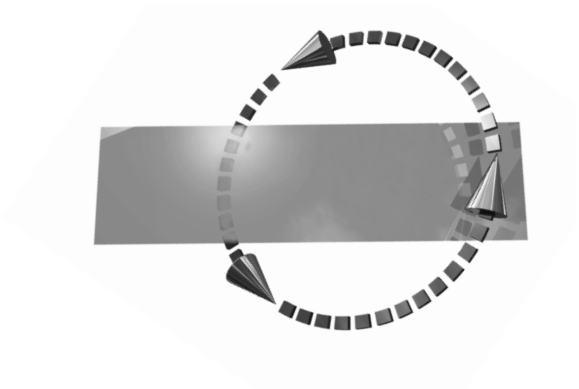

MÄK VR-Link 3.7.2-ngc Release Notes

This release note provides the following release-specific information for MÄK VR-Link Release 3.7.2-ngc:

Document ID: VRL-3.7.2-3-020510

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 3.7.2-ngc. Please contact MÄK if you are interested in purchasing VR-Link for other platforms. Application code must be built with the indicated compilers in order to link to VR-Link libraries.

Table 1: Platforms supported

Platform	Compiler
SGI IRIX6.5	MipsPro7.2.1 C++ compiler (available from SGI)
Sun Sparcstation with Solaris2.7 or later	SPARCompiler C++ version 5.2 (available from Sun)
PC with Red Hat Linux 7.2	GNU C++ (GCC) 3.0.2
PC with Windows NT, Windows 95/98/2000/XP	Microsoft Visual C++ 6.0.

Disk Space Requirements

On SGI IRIX, you need approximately 270 MB for VR-Link files and 60 MB for all documentation files. Other UNIX installations require approximately 50 MB for VR-Link files and an additional 60 MB for documentation files.

On PCs, you need 50 MB for VR-Link files and 60 MB for all documentation files.

Online Help Browser Requirements

The online help system for the FOM Mapper Builder is an HTML-based system. To view online help, you must have a web browser installed that supports Javascript and Java. Java and Javascript must be enabled.

Backward Compatibility

VR-Link 3.7.2-ngc is highly compatible with VR-Link 3.7.1-ngc. For example, the *talk* and *listen* example programs from VR-Link 3.7.2-ngc and previous versions compile and link unmodified against VR-Link 3.7.2-ngc.

Network Compatibility

HLA only

VR-Link 3.7.2-ngc is compliant with:

- RPR-FOM 0.5, 0.7, 0.8, 1.0, and a subset of 2.0 (draft 6)
- MÄK RTI 1.3.6-ngc
- DMSO RTI NG v1, v2, v3.0, v3.1, v3.2, and 4.0 (but not 5.0).

The linux 7.2 verison of VR-Link is not compatible with the DMSO RTI. It runs only with MÄK RTI 1.3.6-ngc for gcc 3.0.2.

DIS only

This release supports DIS 2.0.4, IEEE 1278.1, 2.1.4, and IEEE 1278.1a, and can therefore interoperate with DIS applications of any of these versions.

RPR FOM Versions Supported

VR-Link 3.7.2 has built-in support for versions 0.5, 0.7, 0.8, 1.0, and 2.0 of the RPR FOM. It also supports VR-Link's ability to support alternative FOMs through the FOM Mapper. By default, VR-Link 3.7.2 uses RPR FOM 1.0.

If you want to use a version of the RPR FOM other than 1.0, pass the version number (0.5, 0.7, 0.8, or 2.0) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExerciseConn* constructor. Also, make sure you are using a federation execution name that corresponds to the right FED file. For example:

```
DtExerciseConn conn("VR-Link20", "MyApp", new DtRprFomMapper(2.0));
```

VR-Link examples like *f18* and *hlaNetdump* have a command line option, `-f`, that you can use to choose an RPR FOM version by passing either `vlRPR05`, `vlRPR07`, `vlRPR08`, or `vlRPR20` as the argument to `-f`.

RTI Support

VR-Link 3.7.2-ngc comes with the MÄK RTI 1.3.6-ngc, which is link-compatible with DMSO RTI NG. VR-Link 3.7.2-ngc has been tested with DMSO RTI NG v3.2 and v4.

The Linux 7.2 version of VR-Link 3.7.2 (which was compiled with gcc 3.0.2) runs only with MÄK RTI 1.3.6-ngc for gcc 3.0.2. It has been tested only with this version.

VR-Link 3.7.2 may work with older versions of the MÄK RTI, but we recommend using it with release 1.3.6-ngc.

If you use the MÄK RTI, remember to make sure that VR-Link can find your FED file, and optional *rid.mtl*. Put the FED file in the directory from which you are running, or set the environment variable RTI_CONFIG to the directory that contains it. See *MÄK RTI Reference Manual* for a list of the options for specifying the location of the *rid.mtl* file.

Library Names

VR-Link 3.7.2 expects library names that match those of DMSO RTI NG v4. If you run VR-Link 3.7.2 with DMSO RTI NG v4, both the UNIX and PC versions work without any modification. If you want to run VR-Link 3.7.2 with DMSO RTI NG v1 or v2, you must make the following changes:

- For UNIX, you must use the libraries *libRTI-NGv2.so* or *libRTI-NGv1.so*, which are supplied by MÄK in the *./lib* directory. These libraries correspond to DMSO RTI NG v1 and v2. To use VR-Link 3.7.2 with RTI NG v1 or v2, copy the appropriate library to the DMSO RTI NG *lib* directory and rename it *libRTI-NG.so*. (You might want to rename the v3.x *libRTI-NG.so* library first.)
- For PCs, rename the libraries that come with RTI NG v1 or v2 to the names used by the RTI NG v4 libraries (*libRTI-NG.lib* and *libfedtime.lib*).

New Features and Changes

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

- Support for the use of independent VR-Link objects from multiple threads
- Support for both big and little endian wire formats.
- Support for more 'user-friendly' initialization
- Revised *MÄK VR-Link Developer's Guide*.

Using Independent VR-Link Objects from Multiple Threads.

VR-Link has been modified so that you can use independent VR-Link objects in multiple threads. In general, these changes should be completely transparent to VR-Link users. If you are using VR-Link in multi-threaded applications, note the following:

- Static variables that are used for initialization have remained static (that is, global) and should be treated with care if their values might be modified concurrently by multiple threads.
- License management communication with the license server can become confused if two objects try to check out licenses at the same time. This will result in the following error:

  ```
  Server message checksum failure (-60,147)
  ```

  To solve the problem, stagger the creation of the conflicting objects by calling DtSleep().

- The organization of *DtGeodeticCoord* and *DtUtmCoord* have changed so that you are no longer restricted to using only one map datum or UTM reference point at a time. The global functions DtSetMapDatum(), DtCurrentMapDatum(), and DtUtmInit() have not changed. Using these functions should not result in any conflicts even when multiple threads are sharing the same reference ellipse and reference point. However, when multiple reference data need to be used simultaneously, an object representing the reference ellipsoid or UTM reference point can be passed to the coordinate's constructor. The given object will then be used for any coordinate conversions instead of the defaults.

Two Classes Added

Two classes were added to VR-Link to make this change possible. The first, *DtReferenceEllipsoid* (defined in *refEllipsoid.h*), is used to represent the shape of the Earth. It is initialized using a *DtMapDatum* instance (defined in *mapDatum.h*). The second class, *DtUtmReferencePoint* (defined in *refUtmPt.h*), is used to represent a UTM reference point. It is initialized using a latitude, longitude, boolean value for enabling the use of standard offsets, and a *DtMapDatum* for the shape of the Earth.

For example:

```
DtDegMinSec latRef = {35.0, 0.0, 0.0, DtNorth};
DtDegMinSec lonRef = {122.0, 0.0, 0.0, DtWest};

DtUseMapDatum(&DtWGS84);
DtUtmInit(latRef, lonRef, 1);  // this will use the global map datum

DtGeodeticCoord geod(DtDeg2Rad(30.0), DtDeg2Rad(100.0), 1000.0);
DtUtmCoord utm(geod);
```

is equivalent to the following code segment:

```
DtReferenceEllipsoid ellipse(&DtWGS84);
DtUtmReferencePoint refUtm(latRef, lonRef, 1, &DtWGS84);

DtGeodeticCoord geod(DtDeg2Rad(30.0), DtDeg2Rad(100.0), 1000.0,
   &ellipse);
DtUtmCoord utm(geod, &refUtm);
```

Support for Big and Little Endian Wire Formats

In rare cases, simulations may be forced to communicate with applications that use little endian rather than big endian wire formats. VR-Link provides a mechanism for setting and inspecting the current wire format at runtime using the functions DtSetNetworkByteOrdering() and DtNetworkByteOrdering() (defined in *NetTypes.h*). Big endian is the default.

To use little endian byte ordering, call:

```
DtSetNetworkByteOrdering(DtLittleEndian);
```

To change it back to the default, call:

```
DtSetNetworkByteOrdering(DtBigEndian);
```

Support for "Friendlier" Initialization

For many applications, the fact that VR-Link exits if it fails during initialization is overly restrictive.

DtExerciseConn's constructor now contains an optional argument that can be used to store the result of initialization. When this argument is non-NULL, the exercise connection will not abort if a non-license management related problem occurs during startup. Instead, the cause of the failure will be put into this variable to be inspected later. Please see *dExerciseConn.h* and *hExerciseConn.h* for more information.

To cleanly check for license management errors, two global functions have been added to give you the ability to check whether a VR-Link or RTI license is available from the license server before they try to create any VR-Link data structures: DtHaveVrLinkLicense() or DtHaveRtiLicense() (defined in *checkLicense.h*). These functions return a DtBoolean instead of calling *DtFatalError* and aborting.

Updated MÄK VR-Link Developer's Guide

MÄK VR-Link Developer's Guide has been updated and reprinted for this release.

Bug Fixes

VR-Link 3.7.2 fixes the following bugs:

- The attribute ProcessID in FOM class EnvironmentProcess has changed to ProcessIdentifier to match the RPR FOM 2.0 drafts
- In EnvironmentProcess, the number of environmental records was being encoded and decoded as a short instead of a long.
- FED file parsing is improved in this version.
- The packet server was broken in the previous release. (patches were available)
- Macro definition for what() caused compile errors on some platforms. It has been removed.
- Event IDs were not always getting encoded properly in HLA.
- DtlObjPtr::null() returned true when the object ptr was not NULL.
- *DtSignalInteraction* copy and assignment operators did not work.
- In DIS, in rare cases, entity accessor methods would infinite loop.
- DtTcpSocket did not work correctly.

Known Problems

- On UNIX, RPR FOM 2.0 spatial attributes have four additional bytes padding at the end of the network structure.
- The implementation of IFF does not match the latest RPR FOM specification.