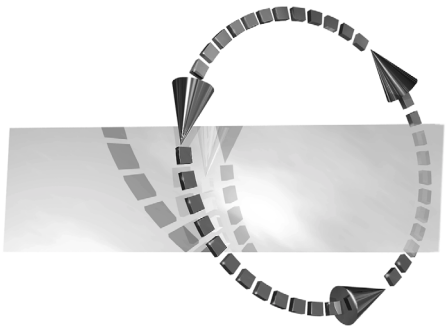

MÄK VR-Link 3.7.1-ngc Release Notes

This release note provides the following release-specific information for MÄK VR-Link Release 3.7.1-ngc:

Document ID: VRL-3.7.1-3-020115

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 3.7.1-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 6.0 (Linux 2.2.16-3)	GNU C++ (g++) distributed with Red Hat distribution. The command: `g++ -v` returns: `gcc version egcs-2.91.66 19990314/Linux (ecgs-1.1.2 release)`
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 3.7.1-ngc is completely compatible with VR-Link 3.7-ngc. There is a high degree of backwards compatibility with VR-Link 3.6-ngc. For example, the *talk* and *listen* example programs from VR-Link 3.6-ngc compile and link unmodified against VR-Link 3.7.1-ngc.

Network Compatibility

HLA only

VR-Link 3.7.1-ngc is compliant with:

- RPR-FOM 0.5, 0.7, 0.8, 1.0, and a subset of 2.0
- MÄK RTI 1.3.6-ngc
- DMSO RTI NG v1, v2, v3.0, v3.1, v3.2, and 4.0.

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 3.7.1 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 3.7.1 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 3.7.1-ngc comes with the MÄK RTI 1.3.6-ngc, which is link-compatible with DMSO RTI NG. VR-Link 3.7.1-ngc has been tested with DMSO RTI NG v1, v2, v3.0, and v3.2.

VR-Link 3.7.1 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 the MÄK RTI Reference Manual for a list of the options for specifying the location of the *rid.mtl* file.

Library Names

VR-Link 3.7.1 expects library names that match those of DMSO RTI NG v3.2. If you run VR-Link 3.7.1 with DMSO RTI NG v3.2, both the UNIX and PC versions work without any modification. If you want to run VR-Link 3.7.1 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 3.7.1 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 v3.2 libraries (*libRTI-NG.lib* and *libfedtime.lib*).

New Features and Changes

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

- Changes to support the MÄK RTI plug-in API.
- Support for DDM.

Support for MÄK RTI 1.3.6-ngc

VR-Link 3.7.1 contains updates to support the new MÄK RTI plug-in API that has been released with MÄK RTI 1.3.6-ngc.

Using DDM With VR-Link in HLA

While VR-Link provides only a small amount of help with DDM over and above what is provided by the RTI, applications can use DDM within a VR-Link application by making most of the relevant calls through the RTI directly. Recall that to make RTI calls using VR-Link, you must obtain a pointer to the RTI Ambassador from the Exercise Connection using DtExerciseConn::rtiAmb().

The following DDM-related RTI calls can be made by an application without having to worry about how these calls will interact with VR-Link:

- associateRegionForUpdates()
- createRegion()
- deleteRegion()
- notifyAboutRegionNotifcation()
- unassociateRegionForUpdates().

Other DDM services require some extra steps to insure that VR-Link and the RTI continue to have a consistent view of the state of the federate. Normally, VR-Link takes care of subscribes for you, using the non-DDM versions of subscribeObjectClassAttributes() and subscribeInteractionClass(). When using DDM, handle subscription and unsubscription yourself by using direct RTI calls subscribeInteractionClassWithRegion() and subscribeObjectClassAttributesWithRegion(). However, make sure that VR-Link does not do its normal non-DDM subscriptions for you first.

VR-Link normally subscribes to interaction classes automatically at the time that you register callbacks for various types of interactions. Specifically, when you call the addCallback() member of the appropriate *DtInteraction* subclass, VR-Link subscribes to the relevant FOM class in addition to registering your callback. When using DDM, disable the automatic subscription by calling setAutoSubscribeFlag(DtFalse) on your *DtExerciseConn*. For objects, the place where VR-Link normally subscribes for you is within the constructor of *DtReflectedEntityList* or other types of *DtReflectedObjectLists*. When using DDM, disable automatic subscription by calling setAutoSubscribeFlag(DtFalse) on your reflected object list.

VR-Link does not automatically unsubscribe to anything, unless you call unsubscribe() on a *DtObjClassDesc* or *DtInterClassDesc*. When using DDM, rather than calling these VR-Link versions of unsubscribe, make direct RTI calls to unsubscribeInteractionClassWithRegion() or unsubscribeObjectClassWithRegion() instead.

Attribute update requests are similar to subscriptions. VR-Link requests updates for all subscribed object classes, and for all discovered object instances. (Update requests are used to make sure that a federate discovers objects that have been registered before it joins the federation execution). With DDM, if you want to make update request:

1. Turn off VR-Link's default requesting behavior by calling DtExerciseConn::setRequestObjectUpdateFlag(DtFalse), and DtExerciseConn::setRequestClassUpdateFlag(DtFalse).
2. Make direct RTI calls to requestClassAttributeValueUpdateWithRegion().

To send interactions, VR-Link has versions of DtExerciseConn::send() and DtExerciseConn::sendStamped() that take pointers to regions as an argument in addition to the DtInteraction itself. When you pass a non-NULL region, VR-Link uses sendInteractionWithRegion() instead of the normal sendInteraction() RTI service call.

The only DDM function not mentioned so far is registerObjectInstanceWithRegion(). VR-Link normally registers objects for you from within a *DtObjectPublisher* constructor, using the non-DDM version of registerObjectInstance. If you want VR-Link to use registerObjectInstanceWithRegion() instead, you must indicate the set of attributes and regions to be passed to the RTI by calling DtHlaObjectManager::setRegionsForRegister() prior to instantiating a publisher. (You can get a pointer to the *DtHlaObjectManager* through the DtExericseConn::hlaObjMgr() function.) At the time that a publisher is being instantiated, we will use the values passed to the most recent invocation of setRegionsForRegister(), so if you are registering several objects, you may be calling setRegionsForRegister() several times. To restore VR-Link to the default state, where subsequent objects are registered without DDM, call DtHlaObject-Manager::setNoRegionsForRegister().

Bug Fixes

VR-Link 3.7.1 fixes the following bugs:

- There was a bug in decoding the following interactions when the interaction contained no variable datum records:
  - EventReport
  - ActionRequest
  - ActionResponse
  - Comment
- The processing of spaces in the FED file did not recognize carriage returns as white space. A file that contains space information and has just carriage returns would not be processed properly. FED files with just carriage returns can now be processed.
- Fixed a crash that occurred during automatic response to synchronization points.
- Typographical errors in *VR-Link.fed* and in *iffSR.h* were corrected. The text “Tcas1” was changed to “TcasI”.
- In DIS, multiple emitter systems on a single entity now work.

Known Problems

- On Linux, if you are running MÄK products and using the MÄK RTI, they must all have been compiled against the same version of VR-Link.
- On UNIX, RPR FOM 2.0 spatial attributes have four additional bytes padding at the end of the network structure.
- In HLA, the marking text attribute for aggregates is longer than specified by the RPR FOM.
- The implementation of IFF does not match the latest RPR FOM specification.