VR-Link 3.9.6 Release Notes

This release note provides the following release-specific information for VR-Link Release 3.9.6:

Document ID: VRL-3.9.6-3-051110

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 3.9.6. 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 IRIX 6.5	MipsPro7.3 C++ compiler (available from SGI)
Sun Sparcstation with Solaris 2.7 or later	SPARCompiler C++ version 5.2 (available from Sun)
PC with Red Hat Linux 9.0 Red Hat Enterprise Linux Workstation 3.0 (use 9.0 build),	GNU C++ (GCC) 3.2
Red Hat Fedora Core 3	GNU C++ (GCC) 3.4.4
PC with Windows 2000/XP	Microsoft Visual C++ 6.0 SP5 Microsoft Visual C++ 7.1

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.

Compiler Compatibility on Windows

MÄK provides versions of product releases that have been compiled with Microsoft Visual C++ 6.0 and 7.1. When you run MÄK products together, for example, the VR-Link and the Stealth, we strongly recommend that you run versions compiled with the same compiler. Mixing products compiled with different versions of the compiler can result in program instability.

Patching the Microsoft Visual C++ 6.0 Compiler

If you use Microsoft Visual C++ version 6.0, you must install a patch to a standard header file, in order to successfully build applications for HLA 1.3 or 1516. The patch comes from Dinkumware, Ltd (the company that developed the Standard C++ Library header files for Microsoft), but for convenience, we are distributing it with VR-Link releases.

The patch fixes a bug that would cause applications to crash if they pass std::sets or std::maps from an application to a DLL or vice-versa. For details about the patch go to http://www.dinkumware.com/vc_fixes.html.

To install the patch:

1. In the top level VR-Link directory, is a file called *XTREE-msvc++patch*. Rename this file *XTREE*.
2. Make a back-up version of the original *XTREE*.
3. In the Visual Studio *include* directory (for example, *C:\Program Files\Microsoft Visual Studio\VC98\Include*), replace the original version of *XTREE* with the patched version.

Backward Compatibility

VR-Link 3.9.6 is highly compatible with 3.9.5. For example, the *talk* and *listen* example programs from VR-Link 3.9.5 and previous versions compile and link unmodified against VR-Link 3.9.6.

Network Compatibility

HLA only

VR-Link 3.9.6 is compliant with:

- RPR-FOM 0.5, 0.7, 0.8, 1.0, and a subset of 2.0 (draft 6, 14, and 17)
- MÄK RTI 2.x
- DMSO RTI NG 6.0.

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 3.9.6 has built-in support for versions 0.5, 0.7, 0.8, 1.0, and 2.0, drafts 6, 14, and 17, of the RPR FOM. By default, VR-Link 3.9.6 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, 2.0006, 2.0014, or 2.0017) 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-Link20017", "MyApp", new DtRprFomMapper(2.0017));
```

VR-Link examples like *f18* and *hlaNetdump* have a command line option, `--rprFomVersion`, that you can use to choose a RPR FOM version, using one of the version numbers listed in the previous paragraph.

RTI Support

VR-Link 3.9.6 has been tested with DMSO RTI NG v6, Pitch RTI 1.3, and MÄK RTI 2.4.

VR-Link 3.9.6 may work with older versions of the MÄK RTI, but we recommend using it with release 2.4.

If you use the MÄK RTI, remember to make sure that VR-Link can find your FED file or FDD file, and optional *rid.mtl*. Put the FED file or FDD 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.

New Features and Changes

VR-Link 3.9.6 has the following changes:

- Support for DDM
- New class for exceptions
- Support for Boost smart pointers
- Additions to *vlutil*
- Changes to *VR-Link20017.fed*
- Miscellaneous changes.

Support for DDM

Data Distribution Management (DDM) has been implemented in VR-Link 3.9.6. There are two ways to use DDM:

- Use the DDM scheme implemented by VR-Link's publishers
- Pass in your own DDM region and manipulate it manually.

The default VR-Link DDM scheme uses a space named BenchmarkGeographicSpace that has two dimensions, X and Y, which correspond, respectively, to longitude and latitude.

VR-Link's Geographic DDM

The *DtEntityPublisher* and *DtAggregatePublisher* are now able to create their own DDM regions on instantiation, and update them based on entity location and velocity. The size of the region depends on velocity. To enable this, you must turn on geographic DDM in the DtExerciseConn, as follows:

```
exConn.setUseGeographicDdm(true);
```

The exercise connection also has a "playbox" that is used to define the minimum and maximum values of latitude and longitude when performing DDM normalization. The region bounds may not exceed those specified by the playbox. The default values cover the entire Earth (-90 to 90 degrees latitude, and –180 to 180 degrees longitude). To change them, for example to the region bounded by 30N-40N latitude and 120E – 130E longitude, do the following:

```
DtGeodeticCoord lowerBound(DtDeg2Rad(30.0), DtDeg2Rad(-130.0), 0);
DtGeodeticCoord upperBound(DtDeg2Rad(40.0, DtDeg2Rad(-120.0), 0);

exConn.setDdmPlayboxLowerBound(lowerBound);
exConn.setDdmPlayboxUpperBound(upperBound);
```

Publishing using Geographic DDM

To publish with a geographic region, the *DtExerciseConn*'s setUseGeographicDDM flag must be set to true. When an entity or aggregate publisher is created, the object is published using a region. This region can be accessed and passed to other publishers, to allow them to publish in the same region. For example, suppose an entity is published with a region. To publish an emitter system in the same region, the entity's region can be passed into the emitter publisher's constructor, as follows:

```
exConn.setUseGeographicDdm(true);

DtEntityPublisher entityPub(...);

DtEmitterSystemPublisher emitterPub(&exConn, 0,
  entityPub.publishingRegion());
```

Reflecting using Geographic DDM

You can create a geographic region and pass it into a reflected list to subscribe to a class with region.

The *DtDDMRegion* and *DtGeodeticRegion* classes in VR-Link are passed into methods using Boost's shared pointer container, since many different objects can hold regions concurrently. The types, DtDDMRegionSP and DtGeodeticRegionSP correspond to DtDDMRegion and DtGeodeticRegion shared pointers.

To subscribe to the region spanning 35 to 40 degrees N and 118 to 122 degrees W, do the following:

```
DtGeodCoord lowerBound(DtDeg2Rad(35.0), DtDeg2Rad(118.0), 0);

DtGeodCoord upperBound(DtDeg2Rad(40.0), DtDeg2Rad(122.0), 0);

DtGeodeticRegionSP region(new DtGeodeticRegion(&exConn));
region->setRegionBounds(lowerBound, upperBound);

DtReflectedEntityList reflEntList(&exConn, region);
```

If the attribute scope advisory switch is enabled in the RTI, the federate will receive attributesOutOfScope callbacks when an object goes out of scope of the region. The object will be removed from the reflected list and rediscovered if it comes back into scope. To enable the advisory switch, do the following:

```
exConn.rtiAmb()->enableAttributeScopeAdvisorySwitch();
```

Notes

- To subscribe to multiple regions, you can create an STL vector of DtGeodeticRegionSPs and pass it into the reflected list constructor.
- Subscription with region is a federate-wide call. In other words, if a *DtReflectedEntityList* is created with one region, and second *DtReflectedEntityList* is created with another region, each list will pick up entities from both regions, because the federate has subscribed to entities in both regions.

Interactions with Regions

VR-Forces also lets you send and receive interactions in regions. The add callback member function of each interaction can take a region shared pointer, and a new member function has been added to *DtExerciseConn* to send interactions with regions.

To subscribe with region:

```
// Set up region bounds
...
DtGeodeticRegionSP region(...);
DtFireInteraction::addCallback(&exConn, fireCb, NULL, region);
```

To send an interaction with region:

```
// Set up region bounds
...
DtGeodeticRegionSP region(...);
DtFireInteraction fireInteraction;
...
exConn.sendStampedWithRegion(fireInteraction, region);
```

Using DDM without VR-Link's Geographic Implementation

You can create any type of region using the *DtDDMRegion* class. You can specify the space (for HLA 1.3) and dimensions, and *DtDDMRegion* will create an RTI region that can be manipulated using normalized values. Each publisher, reflected list, and interaction can now take a *DtDDMRegion*, and will publish or subscribe with whatever *DtDDMRegion* you pass in.

The MC02 Region

The MC02 Federation has developed a more complicated DDM scheme, which has been included as an example in VR-Link, in *./examples/MC02Region*. A Hyperspace routing space is defined that contains three dimensions: subspace, one, and two. Each type of region belongs to some subspace, and the other dimensions depend according to type of region. Example talk and listen applications have also been packaged, to demonstrate how to use the classes. This has been implemented only for HLA 1.3.

DtException Class

The exception class, *DtException*, is the VR-Link exception class. Member functions may throw exceptions under certain conditions, for example, if an object cannot be properly constructed or unexpected input is passed into a method as a parameter. Any member function or constructor that throws an exception will state so in the header file and class documentation. The ostream operator, <<, has been implemented to print exceptions, and the message() member function will return a string that describes the exception.

The *DtVrlException* class has been removed.

Example:

```
// DtDDMRegion may throw exception on creation
try
{
  region = new DtDDMRegion(spaceName, dimensionVector);
}
catch (const DtException& regionException)
{
  DtWarn << "Caught exception: " << regionException << std::endl;
}
```

Boost Smart Pointers

The Boost smart_ptr mechanism has been added to *vlutil*. This has been wrapped in a DtBoost namespace. Include *vlSmartPointers.h* to use the Boost smart_ptr classes. For example, to create a *DtEntityStateRepository* shared pointer:

```
DtBoost::shared_ptr<DtEntityStateRepository> entityStateSP(new
  DtEntityStateRepository);
entityStateSP->setMarkingText("VR-Link");
```

The Boost license statement is in *./doc/BoostLicense.txt*. For details about Boost smart pointers, see http://www.boost.org.

Additions to *vlutil*

The functions, DtTotalPhysicalMemory(), DtAvailablePhysicalMemory(), DtPhysicalMemoryUsed(), and DtPercentageMemoryUsed() have been added to *vlutil* (in *memUtil.h*), to give physical memory information. These functions have not been implemented for Microsoft Visual C++ 6.0.

Changes to *VR-Link20017.fed*

The *VR-Link20017.fed* file was updated in VR-Link 3.9.4 and again for this release. The updated file has been renamed *VR-Link20017-1.fed*. Since many MÄK products use the *VR-Link20017.fed* file distributed with VR-Link 3.9.3, it is still distributed in the *./data* directory of with VR-Link 3.9.6. However, we recommend that any federation that uses RPR FOM 2.0 draft 17 use the updated version of the file. Remember that all federates in a federation must use the same FED file.

Miscellaneous Changes

- The type DtStringSet has been added to *vlStringUtil.h*, which is std::set<DtString>.
- Constructors have been added to *DtReflectedEntity* that allow you to pass in a state repository to be used or pass in a state repository creator, rather than use the static create function.
- The following changes have been made to *vlMath.h*:
  - A templated modulus function, DtMod, has been added.
  - A greatest common divisor function, DtGCD, has been added.
  - A least common multiple, DtLCM, has been added.
- In DIS, in the *DtEmitterSystemPublisher*, modifiers have been added for the beam azimuth and beam elevation thresholds.
- In DIS, in the *DtRadioTransmitterPublisher*, modifiers have been added for the antenna location, and antenna orientation thresholds.
  A moving threshold has also been added to the *DtRadioTransmitterPublisher*. A static flag has been added to turn off this functionality. The radio publisher will use the velocity of the host it's attached to, in order to determine if it's moving or not. If it is, a default threshold of 2 seconds will be used. Otherwise, a default threshold of 5 seconds will be used.
- In the DIS netdump utility, a `--raw` flag has been added as a command line option to run in raw mode. Output like nethex will be printed with each PDU received, along with the PDU # received and header information. This is expected to replace nethex in the future.
- The HLA 1516 *DtExerciseInitializer* and *DtVrlApplicationInitializer* now use *VR-Link.xml* as the default FED file, rather than *VR-Link.fed*.
- In *mtlenv.h*, the macro MASK has been changed to DtMTL_MASK.
- In *genericSR.h*:
  - The member function attrs() (for both HLA 1.3 and 1516) is no longer inline
  - In the member function setValue(handle, buffer, size), buffer is now const
- The *VR-Link20017.fed* file changed between VR-Link 3.9.3 and 3.9.4. Objects and interactions were added to make it closer to RPR FOM draft 17. However, none of the encoding has changed.

Documentation Updates

The printed version of *VR-Link Developer's Guide* has been updated and reprinted for this release.

Bug Fixes

VR-Link 3.9.6 fixes the following bugs:

- The snipBack() member function of *DtString* has been modified to handle searches for the character '\0' in a manner consistant with the other *DtString* methods such as snipFront, size, and count. The null terminator is not considered to be an element of the string, and so searches for it should fail. For example, count('\0') will return 0, and snipFront('\0') and snipBack('\0') will return false and leave the string unmodified.
- Previously, the Change Indicator field of articulated parts data was sent as 0 on each update. Now, the Change Indicator is incremented with each update of an articulated parts parameter.
- In HLA, an emitter system was updated before an emitter beam, resulting in a reflection of the emitter system with beam data from the previous frame. Now, the beam is updated before the system, so that the reflected system will have the most recent beam data.
- The functions DtHaveVrlinkLicense() and DtHaveRtiLicense() previously caused the license finder GUI to pop up if a license was not found. They now return false without the GUI enabled.
- In *disEnums.h*, the enumeration DtInteriorLightsOn was previously 28 and DtExteriorLightsOn was 29. DtExteriorLights is now 28 and DtInteriorLights is 29.
- The lookup() method of *DtReflectedEntityList* by entity ID previously always returned NULL. This has been fixed.
- The method DtVector::magnitudeIsZero(double) now works with tolerance of less than 1.0.
- The function DtStringToInetAddr() has been modified to resolve hostname strings, in addition to IP strings.
- A bug was discovered in HLA 1516, which significantly affected performance. This has been fixed.

Known Problems

This release of VR-Link has the following known problems:

- VR-Link correctly encodes and decodes EnvironmentalProcess objects for RPR FOM 2, Draft 14. However, the RPR FOM specifies a way of encoding which prevents VR-Link from correctly decoding modified or custom Environmental-Records. Unfortunately, many MÄK products built with VR-Link use customized EnvironmentalRecords. If you use these products or if you use customized records in your applications, please do not use the RPR FOM 2 Draft 14 FOM Mapper. This is not a problem in other FOM versions, and it has been resolved in future versions of the FOM.
- The RTI 1516 API specifies that all strings passed to and from the RTI are handled as wide strings. VR-Link (and the MÄK RTI) store strings internally as narrow strings. This means that true multibyte wide strings used by RTI Federates may not be handled correctly when received by VR-Link. VR-Link handles name reservation wide strings correctly, however, as this is a 1516-only function. VR-Link provides conversion functions from Wide To Narrow/Narrow to Wide string conversion in *vlStringUtil.h*.