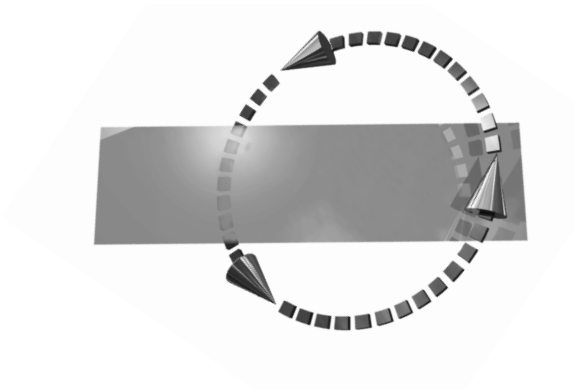

MÄK VR-Link 3.9.1 Release Notes

This release note provides the following release-specific information for MÄK VR-Link Release 3.9.1:

Document ID: VRL-3.9.1-3-040129

Systems Supported

Table 1 lists the platforms currently supported by MÄK VR-Link 3.9.1. 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	MipsPro7.3 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 8.0 and 9.0	GNU C++ (GCC) 3.2
PC with Windows NT/2000/XP	Microsoft Visual C++ 6.0 SP5 Microsoft .NET

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. Javascript must be enabled.

Backward Compatibility

VR-Link 3.9.1 is highly compatible with VR-Link 3.8.x-ngc. For example, the *talk* and *listen* example programs from VR-Link 3.8-ngc and previous versions compile and link unmodified against VR-Link 3.9.1.

Network Compatibility

HLA only

VR-Link 3.9.1 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 2.1
- DMSO RTI NG 4.0, and 6.0 (but not 5.0). The Linux version is not compatible with DMSO RTI NG.

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 3.9.1 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 3.9.1 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, 20006, 20014, or 20017) 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-Link20006", "MyApp", new DtRprFomMapper(20006));
```

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 a RPR FOM keyword as the argument to `-f`. The keywords for specifying different versions of the RPR FOM with the `-f` startup option are:

- RPR FOM 0.5: `vlRPR05`
- RPR FOM 0.7: `vlRPR07`
- RPR FOM 0.8: `vlRPR08`
- RPR FOM 1.0: `vlRPR10`
- RPR FOM 2.0 draft 6: `vlRPR20006`
- RPR FOM 2.0 draft 14: `vlRPR20014`
- RPR FOM 2.0 draft 17: `vlRPR20017`.

RTI Support

VR-Link 3.9.1 has been tested with DMSO RTI NG v6 and MÄK RTI 2.1.

The Linux version of VR-Link 3.9.1 is not compatible with the DMSO RTI (because the DMSO RTI does not support Red Hat Linux 8.0.) If you need a Linux version of VR-Link that is compatible with the DMSO RTI, contact us at support@mak.com.

VR-Link 3.9.1 may work with older versions of the MÄK RTI, but we recommend using it with release 2.1.

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

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

- Support for FlexLM 9.2
- DtBoolean, DtTrue, and DtFalse have been deprecated
- Improvements to byte-swapping routines
- Support for RPR FOM 2.0 draft 17
- Changes to publishers and reflected lists
- Two new functions in the *matrix* library.

Support for FlexLM 9.2

VR-Link 3.9.1 uses FlexLM 9.2. You must replace the files in the flexlm directory on your license server with the new files.

If this is the first time you are installing a MÄK product, just follow the steps for installing license management that are in *VR-Link Developer's Guide*.

To update your *flexlm* directory:

1. If the license server is running, shut it down with *lmdown*.
2. Back up the files in the *flexlm* directory on the license server (*C:\flexlm* on Windows, typically */usr/local/flexlm* on UNIX). (For example, to *flexlm.old*.)

3. Copy all of the files in *vrlink3.9.1/flexlm92* to the *flexlm* directory on the license server.
4. Copy your license files (*.lic* or *.dat*) from the backup directory to the *flexlm* directory.
5. Start the license server.
6. Run a licensed MÄK product and verify that it is checking out a license.

FlexLM is backwards compatible, so you do not have to change your license file, and MÄK applications built with earlier versions of FlexLM will continue to work with FlexLM 9.2. If you have any problems or questions, contact MÄK support at support@mak.com.

DtBoolean, DtTrue, and DtFalse Have Been Deprecated

DtBoolean, DtTrue, and DtFalse have been deprecated in VR-Link. VR-Link now uses the C++ keywords bool, true, and false. DtBoolean, DtTrue, and DtFalse are defined as:

```
#define DtBoolean bool
#define DtTrue true
#define DtFalse false
```

Existing code that uses DtBoolean, DtTrue, and DtFalse will continue to work.

Improvements to Byte-Swapping

When VR-Link sets the network byte order of data, it will try to use operating system calls to byte-swap. On some platforms, these calls may be atomic processor instructions or NO-OPs.

Changes to Publishers and Reflected Lists

DIS publishers and reflected lists have a new static function to set the state repository creator function. This is consistent with the matching HLA classes.

You can pass a state repository that has already been created to a publisher.

New Functions in the *matrix* Library

The *matrix* library has the following new functions:

- DtHeadingFromAToB() – returns the heading along which you would need to be looking to face point B from point A. It expects UTM coordinates.
- DtVelocityFromAToB() – returns the velocity you would need to be traveling at to get from point A to point B in t seconds. This function is independent of the coordinate system.

Documentation Updates

This section lists corrections and updates to *VR-Link Developer's Guide*.

- The example code in section 5.10.1, page 5-41, has an error. The line that reads:

  ```
  DtArtPartList* artParts = esr->artPartsList();
  ```

  should be:

  ```
  DtArtPartList* artParts = esr->artPartList();
  ```

- The text in section 10.1.8, Copying PDUs, is incorrect. The following is correct:

  DtPdu has the copy constructor and assignment operator overloaded. You can copy data as follows:

  ```
  DtPdu pdu1, pdu2;
  pdu1 = pdu2;
  ```

  pdu1 now contains the same data as *pdu2*. The two objects do not, however, share any data objects.

  Classes derived from *DtPdu* do not usually have copy constructors or assignment operators. To ensure that data is copied correctly from one PDU to another, you must use the network representation of the PDU, for example:

  ```
  DtFirePdu pdu1;
  DtFirePdu pdu2((const DtNetFirePdu*)pdu1.netRep());
  ```

  As with the standard PDU class constructors, the *DtPdu* constructor has an optional buffer argument. (See "Using External Buffers in a PDU Class," on page 10-11.)

  The following code is correct in all cases:

  ```
  DtPdu pdu1;
  DtPdu pdu2(pdu1, buffer);
  ```

  To copy a PDU derived from *DtPdu*, using an external buffer:

  ```
  DtFirePdu pdu1;
  int sz = pdu1.length();
  char* bfr = new char[sz];
  DtFirePdu pdu2((const DtNetFirePdu*)pdu1.netRep(), (DtBufferPtr)bfr);
  ```

- In section 11.3.2, "Choosing a Reference Ellipsoid", the sentence that reads:

  "VR-Link has the following *DtMapDatums* pre-defined in *geodCoord.h*

  should read:

  "VR-Link has the following *DtMapDatums* pre-defined in *mapDatum.h*."

Bug Fixes

VR-Link 3.9.1 fixes the following bugs:

- The setModeCode() member function in IFF State Repository did not accept a correct value of 10. Now it does.
- In the 1516 version of VR-Link, publishers now honor attribute update requests.
- In the 1516 version of VR-Link, several obscure crash bugs have 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*.