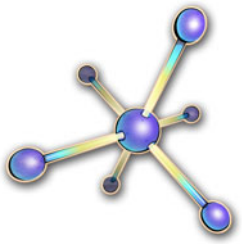

VR-Exchange 1.3 Release Notes

This release note provides release-specific information for VR-Exchange Release 1.3.

Document ID: VRE-1.3-2-070208

System Requirements

This section lists supported platforms and software requirements.

Platforms Supported

VR-Exchange 1.3 supports:

Table 1: Platforms supported

Broker	Platform	Compiler
HLA, DIS	Red Hat Enterprise Linux Workstation 3.0	GNU C++ (GCC) 3.2
HLA, TENA, DIS	Red Hat Enterprise Linux Workstation 4.0	GNU C++ (GCC) 3.4.4
HLA, TENA, DIS	Red Hat Fedora Core 3	GNU C++ (GCC) 3.4.4
HLA, TENA, DIS	PC with Windows 2000/XP	Microsoft Visual C++ 7.1
HLA, DIS	PC with Windows 2000/XP	Microsoft Visual C++ 8.0

External Library Requirements

VR-Exchange 1.3 is built with VR-Link 3.10.1, pThreads 2.70, and Qt 3.3.5. The TENA broker is built with TENA 5.1.1. VR-Exchange 1.3 is not compatible with VR-Link 3.9.x.

TENA users must install TENA middleware and any libraries required by their LROM. HLA users must install an RTI.

Libraries and Binaries Built with Visual Studio 2005

All MÄK products built with Microsoft Visual Studio require the C Runtime Library to function. The C runtime libraries have always been available from Microsoft for download, they are also installed on a user's machine when a Microsoft compiler is installed. The C runtime libraries are not part of the normal Windows installation. For customers who plan to use MÄK products on machines that do not have a compiler installed, MÄK has historically distributed a copy of the C Runtime Libraries with MÄK products. These libraries were put in the *bin* directory used by the MÄK products. MÄK products would then use the libraries in the *bin* directory and customers would not have a problem if copies of the libraries were not already installed.

Unfortunately, with the release of the new C Runtime Libraries required by Microsoft Visual Studio 2005 (MSVC++8.0), the libraries can no longer just be copied into the *bin* directory of an application. The libraries need to be installed correctly into Windows system folders. (The process is actually a little more complicated, a manifest file needs to be created to tell Windows where to find the libraries.)

To accommodate this change, MÄK is distributing the Windows installer for the C runtime libraries with all MÄK products released for MSVC++8.0. The installer is named *vcredist_x86.exe* and is in the base directory of any installed MÄK product.

Running the installer requires Administrator privileges for the machine the installer is run on. MÄK has chosen to not integrate the MÄK installer and the Microsoft installer so as not to require users to have Administrator privileges to install MÄK products. Therefore, if you who do not have a compiler installed, or get error messages like "Software has not been installed correctly, please re-install", you must apply the patch.

For more information see this Microsoft URL:

http://msdn2.microsoft.com/en-us/library/ms235299.aspx

Brokers Provided

VR-Exchange 1.3 includes brokers for:

- HLA 1.3
- HLA 1516 (SISO Standard DLC C++ API for IEEE 1516)
- TENA 5.1.1
- DIS.

Table 2 lists the translators for each broker.

Table 2: Broker translators

Translator	HLA	TENA	DIS
Acknowledge	X	X	
Aggregate	X	X	X
Application Specific	X	X	
Collision	X	X	
Comment	X	X	
Data		X	
Designator		X	
Detonation	X	X	X
Entity State	X	X	X
Environment Process	X		
Fire	X	X	X
IFF	X	X	
Emitter System	X	X	
Radio Receiver	X	X	

Table 2: Broker translators

Translator	HLA	TENA	DIS
Radio Transmitter	X	X	
Set Data		X	
Start	X	X	
Stop	X	X	

New Features and Updates

VR-Exchange 1.3 has the following new features:

- A DIS broker.
- Support for the Microsoft Visual C++ 8.0 compiler.
- Additional object and interaction support for HLA 1.3 and 1516.
- Additional object and interaction support for TENA.
- Additional information is displayed in the broker information windows.
- *DtPortalMessage* now handles the following additional data types:
  - DtU8, DtU16, DtFloat32, DtEntityIdentifier, and DtEntityType
  - DtPortalEntityType has been deprecated in favor of DtEntityType
- DtPortalMessageKind has been extended.

DIS Broker

VR-Exchange 1.3 includes a DIS broker. It is intended to assist customers who need simple translation between DIS and TENA or HLA. However, due to the limited scope of the DIS broker, we recommend that customers who need comprehensive translation between DIS and HLA use the MÄK Gateway.

The VR-Exchange DIS broker translates the following objects and interactions:

- Aggregate
- Detonate
- Entity State
- Fire.

For details about configuring and running the DIS broker, please see the updated PDF version of *VR-Exchange User's Guide*.

Updates to the TENA Broker

The TENA broker has the following new translators:

- Acknowledge
- Application Specific
- Collision
- Comment
- Embedded System
- Emitter System
- IFF
- Radio Receiver
- Radio Transmitter
- Start
- Stop.

The TENA broker has the following changes:

- The previously deprecated file *tenaConversions.h* has been removed. This file contained the following deprecated function declarations:
  - DtEntityType convertEntityType(const DtPortalEntityType & value);
  - DtPortalEntityType convertEntityType(const DtEntityType & value);
- The myLastUpdate and myFirstUpdate data members from the class *DtBrokerObject* have been deprecated. They were never used. They are scheduled to be removed in VR-Exchange 1.4.
- VR-Exchange now supports AMO data in the broker's configuration file. The following parameters have been added:
  - **amoName**
  - **amoHostname**
  - **amoHostIPaddress**
  - **amoOperatorName**
  - **amoOperatorTelephoneNumber**.
- The *DtEntityTranslator* class's translator() functions that took the broker as a parameter have been deprecated. They have been replaced with translator() functions that do not have the broker as a parameter.

The HLA Broker

- The HLA broker has the following new translators:
  - Emitter System
  - Embedded System
  - Environment Process.
- The Jammer Beam and Radar Beam translators have been deprecated. They are now part of the Emitter Beam translator.
- You can now specify the federate name for the HLA Broker on the command line with {`-N` | `--federateName`}. You can also specify it in the configuration file with the parameter **federateName**. The Broker's name is no longer used automatically as the federate name.
- The previously deprecated file *hlaConversions.h* has been removed. This file contained the following deprecated function declarations:
  - DtEntityType convertEntityType(const DtPortalEntityType & value);
  - DtPortalEntityType convertEntityType(const DtEntityType & value);
- DtEntityTranslator::myNextEntityId has been deprecated. Its functionality has been moved to DtHlaBroker::nextEntityId() and DtHlaBroker::useNextEntityId().
- The following previously deprecated member functions have been removed from *DtObjectTranslator*: GatewayObjectCbPtr(), releaseGatewayObject(), findGatewayObject(), and createGatewayObject().
- The following functions have been deprecated:
  - DtDetonationInteractionTranslator::createPortalInteraction(DtDetonationInteraction*)
  - DtFireInteractionTranslator::createPortalInteraction(DtFireInteraction*)
  - DtGenericInteractionTranslator::createPortalInteraction(const DtUnknownInteraction*)

  The deprecated functions did not conform to the VR-Exchange standard for createPortalInteraction, which should accept a const reference. They have been replaced with:

  - DtDetonationInteractionTranslator::createPortalInteraction(const DtDetonationInteraction&)
  - DtFireInteractionTranslator::createPortalInteraction(const DtFireInteraction&)
  - DtGenericInteractionTranslator::createPortalInteraction(const DtUnknownInteraction&)
- All translator classes now use two translate() functions to isolate the heart of the work done during translation. One translate() function translates from a portal message to a HLA message. The other translate() function translates in the opposite direction. For example, in *DtCommentInteractionTranslator* the following two functions are used:

```
virtual DtPortalCommentInteraction* translate(
  DtPortalCommentInteraction*, const DtCommentInteraction& );
virtual DtCommentInteraction* translate( DtCommentInteraction*, const
  DtPortalCommentInteraction& );
```

Updates to Broker Information Windows

Broker information windows now display translator debugging status in a new Translators view and whether or not a translator is enabled.

Other VR-Exchange Enhancements

- DtPortalMessageKind has been extended to reflect the new translators.
- VR-Exchange's GUI now allows you to shut down individual brokers. To shut down a broker, right-click a broker and choose Shut Down Broker.
- Broker configuration files can now test for the identity of the operating system before assigning parameter values. For an example, please see the **IromMapper** parameter in *exampleTenaBroker.mtl*.
- The *DtBroker* member functions initPortalConnection(), initFileLogging(), and shutdownWithError() are now virtual.
- The macro DtDEC_ATTRIBUTE_OVERLOAD() was added to *addAttribute.h*. This gives similar functionality as DtDEC_ATTRIBUTE(). Instead of creating another attribute it creates overloaded mutator and accessor functions which use an already existing attribute.
- *pEmitterBeam.h* has been replaced with *pEmitterBeamData.h* and *pEmitterSystemObject.h*. Emitter Beam is now included in Emitter System.
- The previously deprecated member functions encode(), decode(), and getSize() have been removed from *DtPortalMessage* for DtPortalEntityType.
- *DtHlaAttributeValuePair*, in the class *DtVariableLengthData*, has been deprecated. *DtHlaAttributeValuePair* is replaced with *DtAttributeValuePair*. *DtHlaAttributeValuePair* is scheduled to be removed in VR-Exchange 1.4. *DtAttributeValuePair* is the same as *DtHlaAttributeValuePair*. This was a change in identifier name only.
- The class *DtAppBrokerManager* added the following member functions: brokerItemRightButtonPressed(), brokerPopupMenuPressed(), and shutdownBrokerForItem(). Also, myRightClickPopupMenu has been deprecated because its name incorrectly identifies it as belonging only to a right mouse-click action. It has been replaced with myPopupMenu instead.

Documentation Updates

The PDF version of *VR-Exchange User's Guide* has been updated.

- The last section in this release notes document is a brief primer on installing TENA and the MÄK Object Model, which is included as an appendix in *VR-Exchange User's Guide*.
- The `--noGUI` command-line option applies only to Linux.
- The following information supplements section 3.2.2 "Specifying the ID," in *VR-Exchange User's Guide*.

  The following table shows the reserved broker IDs used by VR-Exchange. These are default values; which can be changed via command line or configuration file parameters.

Table 3: Reserved broker IDs

ID	Usage
0	Portal broker identifier
1	Queue manager
2	Reserved
3	Reserved
4	vrfMessageDump broker identifier
5	Reserved

Bug Fixes

VR-Exchange fixes the following problems that were present in previous releases:

- *vrxMessageDump.exe* no longer requires a licence to be checked out.
- References to Entity State within broker configuration files and classes has been corrected. Previous versions incorrectly referenced this as Entity.
- Removed references to generic translation from the TENA Broker. Generic translation is not applicable to the TENA Broker.
- The TENA broker checks the operating system before it loads the file specified in the **lromMapper** parameter. It is no longer necessary for Linux users to change the configuration file from a Windows-centric format to a Linux format.
- DtFilter::matchPattern() now correctly detects more patterns, for example, "text*". This corrects the way the **objectIdFilters** parameter is used with configuration files.
- HLA Aggregate translation now correctly identifies filtered objects.
- HLA Aggregate translation now uses the HLA broker's configuration file parameters **fillOutgoingEntityIDs** and **useMarkingTextForHlaObjectId**.
- TENA Aggregate translation now uses the TENA Broker's configuration file parameter **fillOutgoingEntityIDs**.

- The values the TENA broker's GUI displays for tick rates are now correct.

Known Problems

VR-Exchange has the following known problems:

- In Windows, double-clicking on a broker in the portal view launches the Broker window in the background. You may need to close other windows to bring it to the front.
- The `--version` and `--help` command line options for *vrx.exe*, *hla13Broker.exe*, *hla1516Broker.exe*, and *tenaBroker.exe* do not display any information.
- On Linux you might encounter the following message:

  ```
  VR-Exchange caught exception: DtSharedMemoryPoolManager()
    create has error with shmget: Invalid argument().
  ```

  This error may occur:

  – When the queue manager cannot reserve enough shared memory for the internal message queues. To correct the error do one of the following:

  • Increase the maximum shared memory allocations on your platform. On Linux the root user can do this by modifying the *shmmax* file:

  ```
  echo "33554432" > /proc/sys/kernel/shmmax
  ```

  This number is the maximum number of bytes allocatable in shared memory. Please consult your operating system documentation first.

  • Reduce the number of buckets, or the size of the buckets in the message queues in *VR-Exchange.mtl*.
- Because of write permissions in the */tmp* dirrectory. When VR-Exchange creates shared memory queues, it creates temporary files in */tmp*. The files are called: */tmp/OQ.shared* */tmp/IQ.shared*, and */tmp/CQ.shared*. If those files exist and the person running VR-Exchange does not have permission to open them for writing, VR-Exchange will print out the error. To correct the problem, delete the files.
- On Linux, it is possible to run multiple instances of the Portal on the same machine. This will result in undefined behavior because all instances of VR-Exchange would use the same message queue.

- If you are using the MÄK RTI in lightweight mode, misconfiguring the HLA brokers can cause a feedback loop to occur. A symptom of this is seeing the number of entities rapidly increase, with the same Entity ID. Although this is not a bug, it is mentioned here to help you diagnose this problem should it occur.

  To avoid such situations, when using the RTI in lightweight mode, make sure the first three characters of each HLA Broker's execName are unique, for example:

  ```
  hla13Broker.exe --execName Boston --brokerName Broker1
    --brokerId 6
  hla13Broker.exe --execName Newton --brokerName Broker2
    --brokerId 7
  ```

  Adding the following broker would cause the feedback loop:

  ```
  hla13Broker.exe --execName Newburyport
    --brokerName Broker3 --brokerId 8
  ```

A VR-Exchange/TENA Primer

TENA can be overwhelming for new users. Detailed documentation about TENA is beyond the scope of this document. This section explains the most basic steps for getting up and running with TENA and the VR-Exchange TENA broker. For complete documentation about TENA, refer to its home web site: www.tena-sda.org.

tena-sda.org is a restricted web site. Access must first be granted by the TENA SDA project office. On the home page, click Account Request.

This section explains how to:

- Install TENA Middleware
- Install the MÄK Object Model
- Build the MÄK Object Model Basic Implementation
- Run the TENA broker.

Installing TENA Middleware

To use the TENA broker, you must download and install the TENA Middleware.

This section explains how to download, install, and verify the installation of the TENA Middleware.

Approximatly 2.98 GB of hard drive space is required for building all the required TENA files.

Downloading the TENA Middleware

To download the TENA Middleware:

1. Log into the TENA web site, www.sda.org.
2. On the Welcome page, click the TENA Middleware link.
3. On the TENA Middleware page, click the Middleware Downloads link. The TENA Repository page is displayed.
4. In the Middleware drop-down list, select TENA-v5.1.1
5. In the table of downloads, click the download button for *TENA-xp-vc71-v5.1.1.exe*

Installing the TENA Middleware

To install the TENA Middleware:

1. Close all open windows and running applications.
2. Run *TENA-xp-vc71-v5.1.1.exe*. This is a self-extracting zip file. In the WinZip Self-Extractor dialog box, make sure to select:

   ```
   When done unzipping open: TENA\5.1.1\scripts\post.install.bat
   ```

Verifying the TENA Middleware Installation

To verify installation:

1. Open MS VC++ 7.1.
2. Open and build the solution TENA\5.1.1\examples\SampleApplication\sampleApplication-2003.sln.
3. Run:

   ```
   TENA\5.1.1\bin\xp-vc71\networkNamingService.exe
     -ORBlistenEndpoints iiop://127.0.0.1:50509
   ```

4. Run:

   ```
   TENA\5.1.1\bin\xp-vc71\executionManager.exe
     -ORBdefaultInitRef corbaloc:iiop:127.0.0.1:50509
     -executionname VR-Link
   ```

5. Run:

   ```
   TENA\5.1.1\examples\SampleApplication\sampleApplica-
     tion.exe
     -ORBdefaultInitRef corbaloc:iiop:127.0.0.1:50509
     -executionname VR-Link
   ```

6. You should see 30 “Updating ...” messages.

Troubleshooting the TENA Installation

If you have problems:

1. Shut down all open programs.
2. Open a new command window.
3. Verify that the following environment variables are set:

   - TENA_ACE_VERSION=5.3.1r
   - TENA_BOOST_LIB_VERSION=1_31
   - TENA_BOOST_VERSION=1.31.0a
   - TENA_HOME=C:\TENA (should point to your own TENA installation path)
   - TENA_OMC_VERSION=0

- TENA_OS=xp
- TENA_PLATFORM=xp-vc71
- TENA_TAO_VERSION=1.3.1r
- TENA_VERSION=5.1.1

4. Make sure your PATH includes:

```
%TENA_HOME%\%TENA_VERSION%\bin\%TENA_PLATFORM%;
  %TENA_HOME%\%TENA_VERSION%\scripts;
```

Building the MÄK Object Model

You do not have to install or build the MÄK Object Model to run VR-Exchange. It is required if you want to build a customized TENA broker.

Downloading the MÄK Object Model

To download the MÄK Object Model:

1. Log into the TENA web site, www.sda.org.
2. On the Welcome page, click the TENA Repository link.
3. At the top of the page, click Browse Repository. A list of object models is displayed.
4. Expand the MAK entry.
5. In the table of MAK object models, click MAK-LROM-v2.
6. In the Distributions list, select TENA-v5.1.1 (Figure 1).

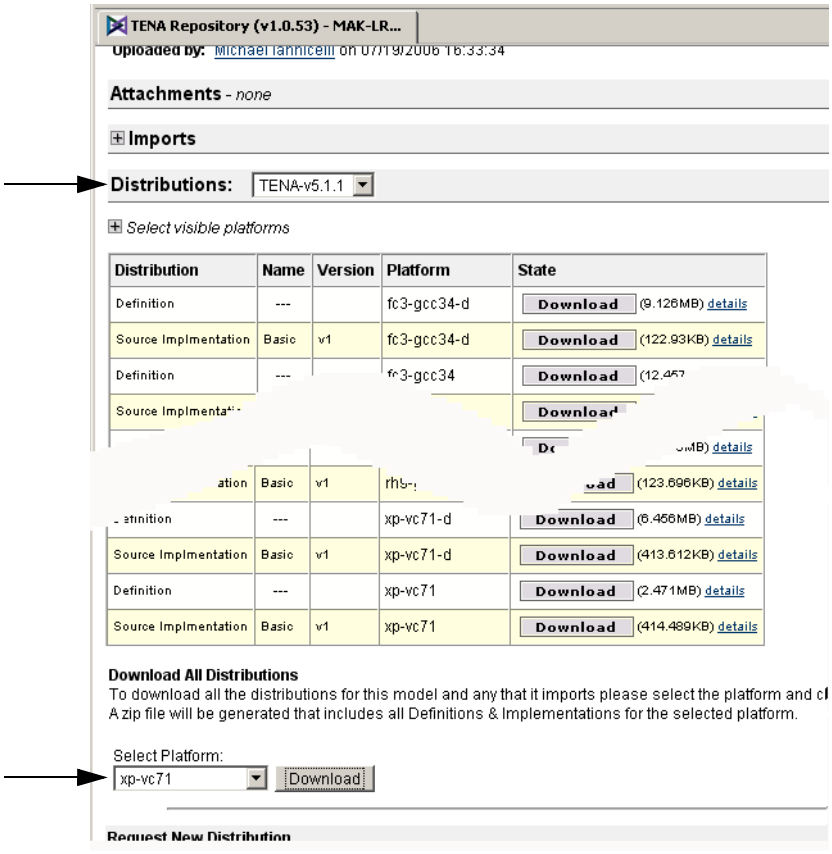

Figure 1. MÄK Object Model download page

7. In the Select Platform list located below the Distributions table, select xp-vc71.
8. Click the Download button next to Select Platform. The distribution is assembled and displayed under the heading Object Model Imports.
9. Click Download Zip File. The file *MAK-LROM-v2-distributions-xp-vc71-v5.1.1.zip* is downloaded.

Installing the MÄK Object Model

To install the MÄK Object Model:

1. Download the MÄK Object Model files as described in "Downloading the MÄK Object Model."
2. Extract all files from *MAK-LROM-v2-distributions-xp-vc71-v5.1.1.zip*. The extracted files are either *.exe* files or zip files.
3. Run all of the *.exe* files. They are self-extracting zip files. Make sure the destination folder is the same as the TENA installation folder.

! When you extract the files, do not select:

```
When done unzipping open: TENA\5.1.1\scripts\post.install.bat
```

Building the MÄK Object Model Basic Implementation

The dynamic libraries are included in the VR-Exchange installation. You do not need to build them unless you build a customized TENA broker.

To build the MÄK Object Model basic implementation:

1. Download the MÄK Object Model files as described in "Downloading the MÄK Object Model."
2. Extract all files from *MAK-LROM-v2-distributions-xp-vc71-v5.1.1.zip*. The extracted files are either *.exe* files or zip files.
3. Unzip each of the zip files that you extracted. Make sure the destination folder is the same as the TENA installation folder.
4. Open MS VC++ 7.1.
5. Open and build *TENA\5.1.1\src\MAK-LROM-v2\MAK-LROM-v2-2003.sln*.

---

i MS VC++ 7.1 has experienced problems with large amounts of text in the Additional Include Directories project setting. If header files cannot be found, reduce the amount of text in this project setting. For example, change all instances of *$(TENA_HOME)\$(TENA_VERSION)\include* to *C:\TENA\5.1.1\include*.

---

6. Create the directory: *TENA\5.1.1\include\impls\MAK-LROM-v2-basic-v1*.
7. Copy *TENA\5.1.1\src\MAK-LROM-v2*.h* (2 files) to *TENA\5.1.1\include\impls\MAK-LROM-v2-basic-v1*.
8. Copy the files in *TENA\5.1.1\src\MAK-LROM-v2\MAK* to *TENA\5.1.1\include\impls\MAK-LROM-v2-basic-v1\MAK*.

9. Copy *TENA\5.1.1\src\MAK-LROM-v2*.lib* and **.exp* (2 files) to *TENA\lib*.
10. Copy *TENA\5.1.1\src\MAK-LROM-v2*.dll* (1 file) to *TENA\5.1.1\bin\xp-vc71*.

Running the TENA Broker

For the TENA broker to communicate with other TENA executions, two TENA services must be running: the Network Naming Service and the Execution Manager.

1. Verify that the following environment variables are set:
   - TENA_ACE_VERSION=5.3.1r
   - TENA_BOOST_LIB_VERSION=1_31
   - TENA_BOOST_VERSION=1.31.0a
   - TENA_HOME=C:\TENA (should point to your TENA installation path)
   - TENA_OMC_VERSION=0
   - TENA_OS=xp
   - TENA_PLATFORM=xp-vc71
   - TENA_TAO_VERSION=1.3.1r
   - TENA_VERSION=5.1.1
2. Make sure that the PATH includes:

```
%TENA_HOME%\%TENA_VERSION%\bin\%TENA_PLATFORM%;
  %TENA_HOME%\%TENA_VERSION%\scripts;
```

3. Start the Network Naming Service by running:

```
TENA\5.1.1\bin\xp-vc71\networkNamingService.exe
  -ORBlistenEndpoints iiop://127.0.0.1:50509
```

4. Start the Execution Manager by running:

```
TENA\5.1.1\bin\xp-vc71\executionManager.exe
  -ORBdefaultInitRef corbaloc:iiop:127.0.0.1:50509
  -executionname VR-Link
```

5. Start VR-Exchange and the TENA broker.
6. For an example of how to start a TENA broker, look at the VR-Exchange file: *./bin/launchTenaExecutionBroker.bat*.