



VR-Exchange

The Interoperability Portal

What is VR-Exchange?

VR-Exchange is a Universal Translator for distributed simulation. It allows assets such as Live, Virtual, and Constructive (LVC) simulations, C4I systems, and after-action-review systems to interoperate, even if those assets are:

Using Local RTI Components from different RTI vendors

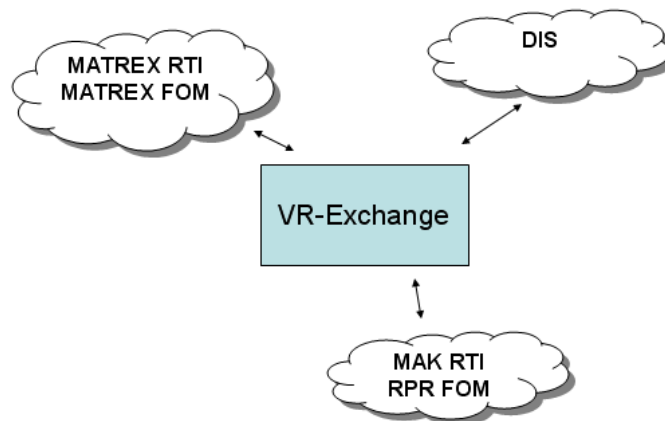
Using different HLA FOMs

Using different communication infrastructures (e.g. DIS, HLA 1.3, HLA 1516, TENA, Link16, etc.)

VR-Exchange can do simple bridging between two federations (simulation environments), or can be used to support a more complex Federation of Federations architecture, where multiple, heterogeneous sites are connected to support large-scale LVC integration.

Is VR-Exchange a Bridge/Gateway?

Yes. VR-Exchange is a software application that allows you to join together several sub-federations into one larger federation. For example, one sub-federation may be using DIS on port 3000, another using the MAK RTI with the RPR FOM, while a third uses the MATREX RTI with the MATREX FOM.



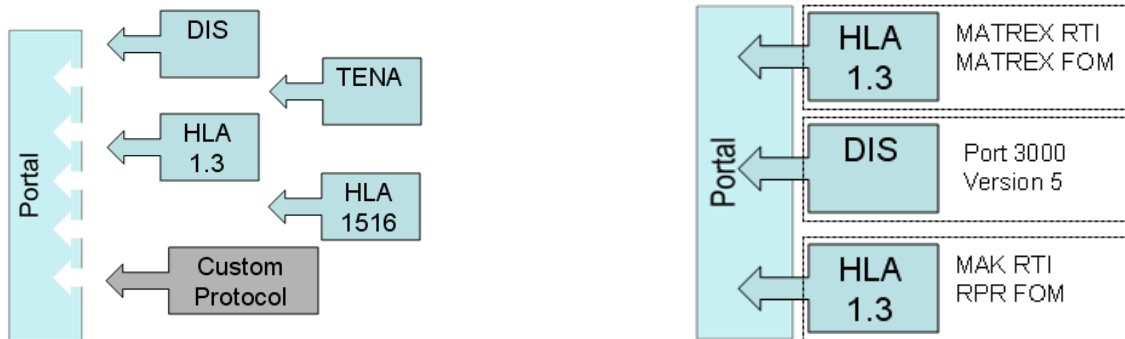
VR-Exchange exchanges all data among the various sub-federations, so that objects and events from one federation can be seen by all the others.

When is VR-Exchange necessary?

In general, bridging is necessary whenever it is not possible to achieve direct interoperability among a set of assets. Perhaps different federates have been developed to different FOMs, and were not developed using a FOM-Agile middleware toolkit like VR-Link. Bridging between the FOMs may be a more feasible solution than changing one set of federates to use the other federates' FOM. Perhaps one set of federates are using HLA 1.3 while others are using HLA 1516, and you are not using an RTI that supports wire-compatibility between the two (such as the MÄK RTI). Bridging may be a more feasible solution than upgrading the HLA 1.3 federates to IEEE 1516. Perhaps different federate developers are used to running with specific RTI implementations, and are unwilling to switch to a common choice. Bridging may be a more politically palatable solution than arguing about whose RTI to use. Perhaps you have some HLA Federates and some TENA Federates that need to play in the same federation. Bridging may be the only way to achieve this goal.

What does the architecture of VR-Exchange look like?

The VR-Exchange architecture consists of a protocol-independent Data Exchange called the Portal with a Broker for each supported protocol. Each Broker is then given a specific configuration to create a Connection. So, if you want to bridge two different RTI's, you would make two Connections which use the HLA 1.3 Broker, but configure each to point to a different RTI's LRC.



Each Connection runs as a separate process, and trades data with the other Connections through the Portal. The Portal creates and maintains a central shared memory pool through which Brokers connect. Although VR-Exchange is made up of several processes, all of the Connections are launched by the Portal, so a user ever only needs to launch a single executable.

The Portal provides a protocol-independent data representation for commonly used object and interaction types – a *lingua franca* for interoperability among protocols. Each Broker is responsible for converting data between its local sub-federation's protocol or object model and the Portals's *lingua franca*.

Does VR-Exchange support many-to-many translation?

Yes. VR-Exchange is not limited to just two Connections at a time. It can be configured with several Connections at once, allowing it to exchange data among three or more

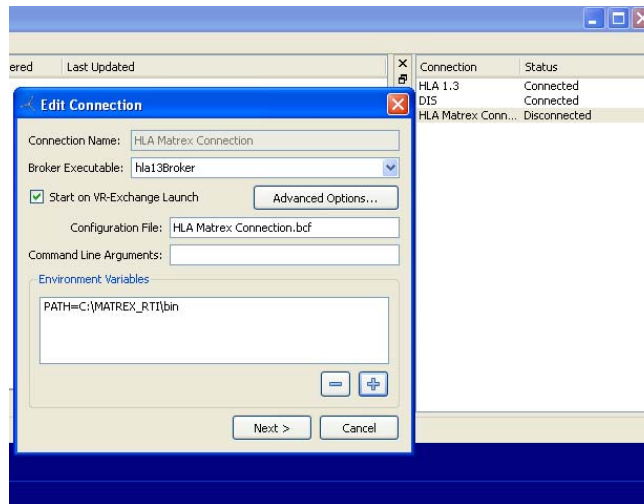
simulation environments. Each Connection passes data that it receives from its own environment to the Portal where it is made available to all of the other Connections.

What protocols are supported out of the box?

VR-Exchange includes Brokers for HLA 1.3, IEEE 1516, DIS, and TENA (the Test and Training ENabling Architecture). It can bridge between different versions of HLA (1.3 and 1516), or between different vendors' RTIs, and can translate between different FOMs. It can bridge between various TENA LROMs, or between TENA, HLA, and DIS. We plan to implement additional Brokers based on customer demand. HLA Evolved (IEEE 1516-200X) and various C4I protocols are leading candidates.

How do I configure VR-Exchange for RTI-to-RTI bridging?

It's simple! Use the intuitive user interface to create two new connections with the HLA Broker. Then, set the system path (under Advanced Options) for each Connection to find the correct RTI LRCs. This configuration should take less than one minute.



When doing RTI-to-RTI bridging, how does VR-Exchange load two RTI implementations at once?

In general, it is not possible to link two different RTI implementations into the same executable. This is because different RTI implementations all implement the same API; if you attempt to link two different RTI DLLs into the same application, symbol name collisions will result. We were able to avoid this problem by designing VR-Exchange as a system of Brokers that communicate through the shared-memory Data Exchange. When used as an RTI-to-RTI bridge, VR-Exchange is configured with two HLA Brokers, each configured with its own RTI DLL. Each Broker is actually a separate executable, so no single executable needs to link with more than one RTI DLL.

How do I configure VR-Exchange for FOM-to-FOM translation?

VR-Exchange's HLA Broker leverages VR-Link's FOM Mapping architecture for FOM-to-FOM translation. Each HLA Connection needs to be configured with a FOM Mapper DLL appropriate to that Connection's local sub-federation. A FOM Mapper provides mappings between the FOM-independent API supplied by VR-Link, and the FOM of

your choice. Like all MÄK products, VR-Exchange ships with a FOM Mapper for the RPR FOM. For other FOMs, you will need to write a custom FOM Mapper, so that VR-Exchange's HLA Brokers (and other MÄK products) can understand your data representations.

If I am just doing RTI-to-RTI Bridging, do I still need a FOM Mapper?

No. If you are using the identical FOM on both sides (or if at least the FOM classes that need to be passed from federation to federation are identical), then you do not need to provide any FOM Mappers at all. This will be the case if you are using VR-Exchange simply as an RTI-to-RTI bridge, and not as a FOM-to-FOM translator. If you are not translating from one FOM to another, then VR-Exchange doesn't need to understand the contents of your updates and interactions – it can just pass them through the Data Exchange from one federation to another in a generic fashion.

Can I use VR-Exchange to translate among different TENA LROMs?

Yes. MÄK's TENA products support "LROM Mapping" – which is analogous in concept to FOM Mapping in HLA. If you configure VR-Exchange with two TENA Connections, each using a different LROM Mapper, you will be able to achieve interoperability between TENA Applications that were built to different LROMs (assuming, of course, that the two LROMs represent similar concepts, and that there is sensible mapping between them.) Please see our separate FAQ on MÄK's TENA Support for more information.

Can VR-Exchange be used as a DIS-to-HLA bridge?

Yes! VR-Exchange is preconfigured to have both a DIS and an HLA Connection right out of the box. All you need to do is install VR-Exchange.

Can VR-Exchange be used as a TENA-to-HLA bridge?

Yes. VR-Exchange includes Brokers for TENA as well as HLA 1.3 and the IEEE 1516 version of HLA. When Connections are made with these Brokers, VR-Exchange supports out-of-the-box translation between HLA and TENA. (Of course, your HLA FOM and your TENA LROM must represent similar enough concepts that a mapping between them is sensible.)

Does VR-Exchange also support DIS-TENA bridging?

Yes! VR-Exchange also includes a DIS Broker, which supports all the commonly used DIS PDUs. Configure VR-Exchange with a DIS Connection and TENA Connection for out-of-the-box translation between DIS and TENA.

What kinds of objects and interactions can VR-Exchange handle?

Well, as mentioned above, if you are not doing data translation, VR-Exchange can handle any arbitrary object and interaction classes out of the box. If you *are* performing data translation, then each Broker needs to convert data from the HLA FOM, TENA LROM, or other protocol on its side of the bridge to the protocol-independent data representation defined by the Data Exchange. The Data Exchange provides a built-in protocol-independent data representation for a wide variety of objects and interactions including:

Objects

- Aggregate
- Designator
- Electromagnetic Emission
- Environment Process
- Gridded Data
- IFF
- Radio Receiver
- Radio Transmitter

Interactions

- Acknowledge
- Collision
- Comment
- Data
- SetData
- Signal
- Start/Resume
- Stop/Freeze

If you have a requirement to pass additional kinds of data across VR-Exchange, the Data Exchange's *lingua franca* would need to be extended to cover the new object and interaction concepts. Extensions can be added either by MÄK or by a user through the API (see below).

Does VR-Exchange have an open architecture, so that I can build custom Brokers?

Yes. VR-Exchange's open architecture means that you can develop custom Brokers for your own protocols. We provide a documented interface (API) to the Data Exchange, as well as sample code that shows how to read and write to our protocol-independent data format. You don't have to write a separate translator between your protocol and every other protocol you need to interoperate with. Once you are able to translate your protocol to the *lingua franca* used by the Data Exchange, your data can be read by any of the other VR-Exchange Brokers, and passed on to their federations.

What the does VR-Exchange API look like?

VR-Exchange provides a base class called DtBroker, which can be subclassed to build new brokers for custom protocols. (The built-in DtHLABroker and DtTENABroker are derived from DtBroker as well.) The DtBroker base class automatically initiates and manages a DtPortalConnection - your connection to the central portal executable, and its shared-memory-based Data Exchange.

VR-Exchange's libPortal library contains a variety of classes that help you pass data in and out of the Data Exchange. DtPortalObjectPublishers allow you to publish objects through the Data Exchange, based on data that your Broker has received from your protocol. DtPortalReflectedObjectManagers manage the objects that are coming through the Data Exchange from other Brokers. You can use DtPortalReflectedObject member functions to grab data about these remote objects, so that you can translate and send the information via your custom protocol. For those who have used MÄK's VR-Link API, you will find these "publisher" and "reflected object" concepts very familiar. The libPortal library similarly provides DtPortalInteraction classes for sending and receiving interactions or events through the Data Exchange.

Does the API allow me to extend VR-Exchange to support new object and interaction types?

Yes. If you need to extend the set of object or interaction classes that VR-Exchange can handle, the VR-Exchange API allows you to do this. You can extend VR-Exchange's *lingua franca* (the protocol-independent data representation defined by libPortal), to add new kinds of DtPortalObjectPublishers, DtPortalReflectedObjects, and DtPortalReflectedObjectManagers. The API uses C++ templates to ease the job of adding new types. You can define a new kind of DtPortalObject in one place, and use template instantiation to generate the publisher, reflected object, and reflected object list for the new type. Once you've extended the Data Exchange in this way, you can extend both the built-in and custom Brokers, so that they can pass the new types through the Data Exchange.

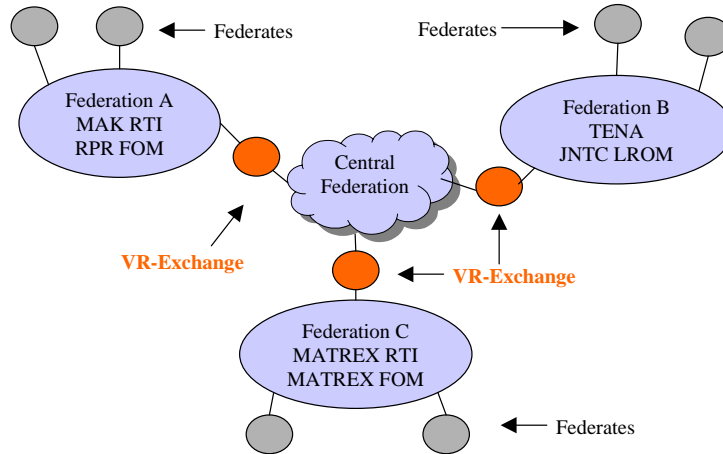
What is the “Federation of Federations” system architecture?

Historically, most distributed simulations have been more or less homogeneous. People have typically put together exercises where everyone used DIS, or where everyone used HLA through a particular RTI implementation and FOM. In those cases where both DIS and HLA federates needed to coexist, a DIS/HLA Gateway has been used. But federations that involved more than one RTI implementation, more than one FOM, or more protocols than just HLA and DIS have been rare.

In the last couple of years, things have changed: Distributed simulations are more often being put together from existing assets, which have been built, tested, and verified against some set of pre-existing protocol choices. There is more integration between Live, Virtual, and Constructive assets. In addition, large exercises are becoming more common, with multiple sites connected over a Wide Area Network. These exercises are widely distributed not only in the sense of geography and network topology; the program management and organizational responsibility is also more likely than in the past to be distributed among many organizations. Each site manager might have his own budget, and might want to make his own decisions about what protocols to use.

The bottom line is that more and more people are thinking of these large distributed exercises as a “system of systems”, or “Federations of Federations” problem. Each site wants to get their “sub-federation” built and tested independently, but they still want to be able to easily integrate with other sites on short time and money. Rather than spending time and money getting all of the sites, and all of the simulation assets to agree on a single architecture, it is often viewed as cheaper and easier to use an “Interoperability Portal” such as VR-Exchange to connect the sites. (This approach is often referred to as a “Mixed Architecture” design).

In many cases, the way this is done is by defining a “central” or backbone federation and requiring that each site use a portal like VR-Exchange to convert from their site's local federation, to the protocol, FOM, and RTI choice dictated by the central federation:



For example, the US Army’s Electronic Proving Grounds’ DTE5/MSDE experiment in September 2005 defined a central federation based on the MÄK RTI and RPR FOM. Each site had the choice of connecting their federates directly to the central federation, or using a Gateway or interoperability portal to join. Several sites used the MATREX RTI and MATREX FOM, for example, but bridged into the central federation using VR-Exchange, which was configured to convert between the MATREX RTI / MATREX FOM and the MÄK RTI / RPR FOM.

The US Air Force Distributed Mission Operations (DMO) program uses a similar “Federation of Federations” system architecture, as does the Canadian DND’s War In a Box Program. (The latter uses MÄK’s VR-Exchange).

What are the pros and cons of using a “Federation of Federations” approach?

From the technical side, getting everyone to agree on a common protocol, RTI, or object model is still usually the best approach. Your federates will be more tightly coupled, and you can take advantage of optimizations that in HLA, for example, only the RTI can provide (based on its knowledge of your overall network topology). For example, you lose some of the benefits of HLA DDM (Data Distribution Management) when going through a bridge. Ownership management and Time Management become more problematic. The MOM (Management Object Model) does not necessarily give you the answers you would expect, because no single RTI knows about all of the federates in the larger exercise. If your desire is really to achieve the “look and feel” of all federates participating in one, unified federation, then we recommend trying to use one, consistent RTI and FOM, if at all possible.

However, there are many situations where a distinct separation between different sets of federates is actually desirable! For example, in a “hierarchical federation”, a group of federates might use an intra-simulator FOM to exchange data among components of a single vehicle simulator, but might want only “external” data such as position, velocity, and orientation to be available to other applications or sites. Perhaps different sub-federations are running at different levels of security, but still need to share some common data. Perhaps you would like to implement filtering at the site level, so that you have a single place from which to control what data can go between sites. Perhaps you

want to be able to bring a running sub-federation in and out of a larger exercise. For these situations, a “Federation of Federations” system architecture is often the best solution.

Then there are the organizational benefits. As described above, the federation of federations approach often makes large-scale exercises easier to manage. Sub-federations on each site are free to choose their own interoperability architecture and object model, and can be independently developed, tested and verified. The integration effort to bring the sub-federations together becomes more manageable.

The bottom line is that if you feel that these improvements to the *process* of setting up an exercise outweigh the technical advantages of putting together a single large-scale federation on top of a common RTI and object model, then the Federation of Federations or “Mixed Architecture” approach (based on an interoperability portal like VR-Exchange) is the right choice.

Does VR-Exchange support JFCOM’s LVC Bridge, or LVCDT concept?

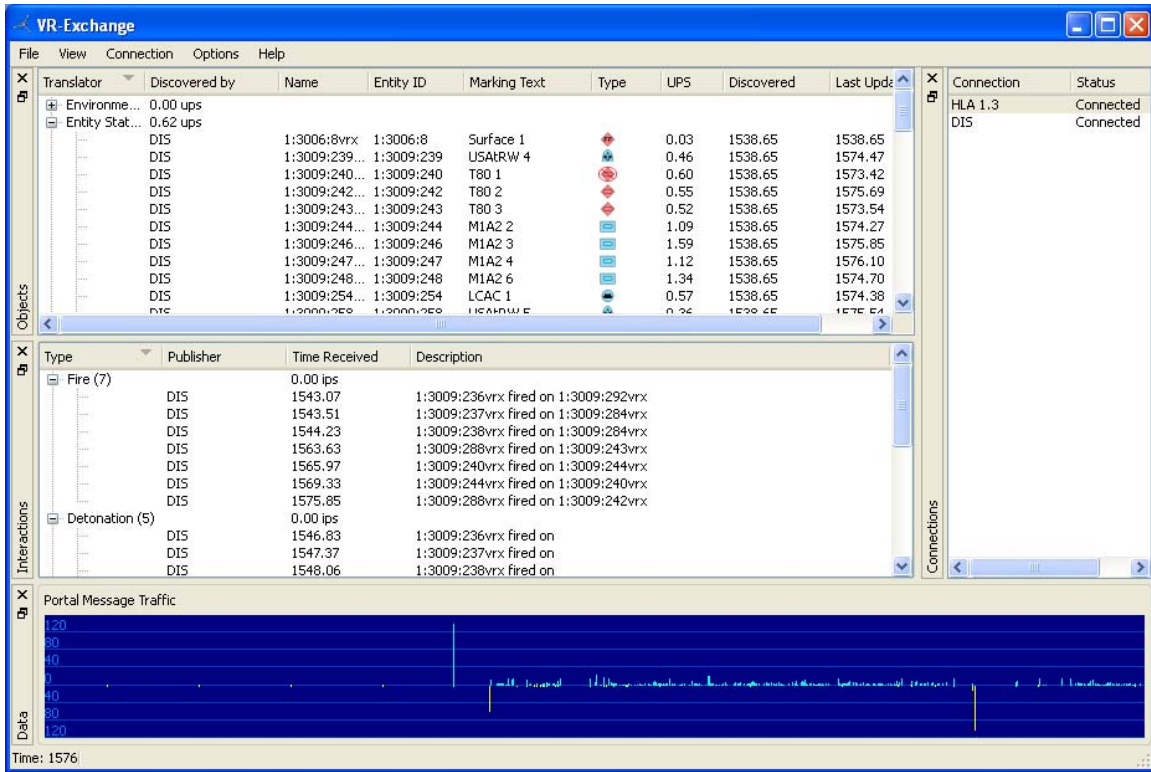
VR-Exchange provides an architecture and implementation that are directly in line with JFCOM’s Live, Virtual, and Constructive Data Translator concept. VR-Exchange has built-in support for HLA and TENA; its architecture is flexible enough to support Brokers for Link-16 and various C4I protocols; and the ability to support simulation-to-C4I interoperability was one of VR-Exchange’s key design criteria.

What kind of impact does VR-Exchange have on performance?

Performance will always be best if you can get all of your federates to agree on a single interoperability protocol, RTI implementation, and object model. Any time you put a bridge between a set of federates, there will always be *some* impact on performance. However, like all MAK Products, VR-Exchange was built with high-performance as a key requirement. In general, VR-Exchange adds only a small amount of latency, due to the additional network hop. Because of VR-Exchange’s multi-threaded, shared-memory-based architecture, the amount of time a piece of data actually spends passing through VR-Exchange is typically negligible. VR-Exchange can easily keep up with exercises that contain thousands of objects (assuming the RTI or TENA Middleware you have chosen can keep up!)

Does VR-Exchange provide a GUI, so I can see what is happening inside?

Yes. VR-Exchange is designed to be fully accessible via a GUI. Users can configure all aspects of VR-Exchange through the GUI, as well as understand what’s going on in each of their connected networks.



What does the future hold for VR-Exchange?

In the future we plan to add more Brokers to support additional built in protocols. As mentioned earlier, HLA Evolved and some of the C4I protocols are also some likely candidates. This may also include distributing FOM Mappers or LROM Mappers for other common object models, so that VR-Exchange can perform out-of-the-box translation among common combinations.

As with all MÄK products, we try to make feature prioritization decisions based on customer feedback. If there are features or extensions that are important to your program, please let us know.