



# The MÄK High-Performance RTI

Are you concerned about the performance of your HLA federations? Worried that your RTI won't be scalable enough to work in large, WAN-based federations, or won't be supported on all of the platforms that you need? Frustrated with the notion that your RTI is a black-box, making connectivity problems difficult to debug? Confused about why your RTI isn't easier to configure and use? Wondering how you will be able to support HLA 1.3, IEEE 1516-2000, and IEEE 1516-2010 (also known as HLA Evolved) federations? Considering writing your own RTI, because you need one that is tailored to your program's specific needs?

If so, then you probably aren't yet using the MÄK High-Performance RTI, the only RTI designed specifically to address all of these concerns. Consistently proven by third party studies to be the most efficient RTI available, the MÄK RTI is also extremely portable, easy to debug, install, configure and use, and backed by MÄK's renowned technical support team. The MÄK RTI offers GUI-based connection configuration, and a web-based Diagnostic GUI that allows you to examine the RTI's internal state from any web browser. And only the MÄK RTI provides a plug-in API that allows you to extend or alter the functionality of a proven RTI. Read on to find out how the MÄK RTI can reduce the cost, risk, and schedule of developing and maintaining your HLA federation.

**Download and start using the MÄK RTI for free! The MÄK RTI can be used in federations of up to two federates, free of charge, without even requiring a license key: <http://www.mak.com/products/rti.php>**

## What is an RTI?

An RTI, or Run-Time Infrastructure, is a required component of any HLA (High Level Architecture) federation. According to the rules of HLA, various simulation applications, known as federates, use the RTI to exchange simulation data.

The HLA Interface Specification, part of the HLA Standard, dictates the set of functionality that federates can expect of an RTI, and provides API mappings for various programming languages, including C++ and Java. An RTI consists of a set of libraries that implement this standard API, along with supporting tools required by the implementation (for example, a central server application known as an RTI Executive or rtiexec). Federates communicate by making calls to RTI functions through the standard API.

RTI functionality falls into eight categories: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management, Data Distribution Management, Support Services, and the Management Object Model (MOM).

## Who is using the MÄK RTI?

These days, it seems like almost everyone. We've sold many thousands of licenses of the MÄK RTI around the world. Including site licenses and title licenses, well over ten thousand federates use the MÄK RTI. Many HLA product companies such as FATS and OPNET distribute the MÄK RTI with their HLA products, and many more, such as AGI, Calytrix, ASTi, and Boston Dynamics, specifically test their products against the MÄK RTI before each release.

It is rapidly becoming the de-facto standard in the industry for both HLA 1.3 and IEEE 1516 federations, both large and small. Some key programs that have chosen the MÄK RTI:

- US Air Force Joint Strike Fighter program
- US Air Force Distributed Mission Operations
- NATO's First WAVE exercise
- US Navy DDG 1000 program
- Canada's Department of National Defense War in a Box program
- US Marine Corps Tactical Environment Network
- US Army Virtual Proving Ground SEIT Demonstration
- Netherlands' TACTIS program
- Australian Defence Simulation Office Joint Simulation Capability program

In Canada, the Department of National Defense (DND) chose the IEEE 1516 version of the MÄK RTI (after a formal evaluation and selection process), for the War in a Box project – part of the Canadian Advanced Synthetic Environment (CASE) project.

In Australia, the Australian Defence Simulation Office (ADSO) chose to standardize on the MÄK RTI for High Level Architecture (HLA) networking for the Joint Simulation Capability, in conjunction with the Australian Army, Navy, Air Force, and Defence Science Technology Organization (DSTO).

In Europe, the MÄK RTI was the backbone of Exercise FirstWave – a major event in early 2005 that included sixteen sites in seven different countries, on both sides of the Atlantic. The MÄK RTI was also used in the EADS GeneSyS project. Various divisions of European companies such as Thales, Alenia Marconi, BAE and Saab have all standardized on the MÄK RTI.

And in Asia, the MÄK RTI is being used by NEC on the JDSS (Japan Defense Simulation System) Program.

In fact, the MÄK RTI is used by *all* of the top ten defense contractors in the world.

### **How long has the MÄK RTI been available?**

The MÄK RTI was first released in June 1998, making it the first commercially developed RTI. It has been in continuous use ever since, giving it a longer history than *any* other RTI available.

The MÄK RTI was originally developed in response to concerns expressed by the real-time simulation community about the performance of early RTIs. We initially focused on meeting the needs of that segment of the HLA community, providing an RTI that kept latency, bandwidth, and CPU usage to an absolute minimum. At the time, the MÄK RTI implemented only those services that were typically required by real-time federations. But over time, DDM, Time Management, and MOM were added, and by early 2002, the MÄK RTI was a full implementation of the entire HLA Interface Specification, Version 1.3.

### **Why is the MÄK RTI so Fast?**

To us performance is not an afterthought; performance is part of the design. Configuration file parameters allow you to disable services that you do not need. If you do not have Time Management or DDM enabled, for example, we automatically use a more compact message format. Obviously, though, if you do use the additional services, extra bookkeeping information needs to be exchanged. In general, latencies are not affected by additional services, since our communications strategies have not significantly changed since the original subset RTI was released. We have also taken great pains to make sure that you do not pay a penalty for unused services in terms of computation time. In general, we check whether the advanced services are enabled, and revert to our original, compact algorithms if they are not. For example, if MOM is not enabled, we avoid collecting the information that would be necessary to send MOM updates.

For further information about the performance of the MÄK RTI, please ask for a copy of a paper we have published, entitled The MÄK High-Performance RTI: Performance By Design.

### **Is the MÄK RTI *only* suitable for real-time simulations?**

No. While the MÄK RTI's low-latencies and efficient use of network and computational resources make it *particularly* well-suited to real-time simulations, the MÄK RTI is a complete implementation of the HLA Interface Specification. It includes all of the Time-Management services necessary for use in non-real-time environments, including time-stepped, or event-based simulations. Further, our focus on performance has carried through to our implementation of time-management, and other RTI services. After all, it's not just real-time simulations that care about the performance and scalability of an RTI.

### **Does the MÄK RTI support any FOM?**

Absolutely, like any RTI, the MÄK RTI will read your FED file (or FDD file for IEEE 1516) to obtain the information it needs about your federation's FOM – names of classes, attributes and parameters; transport and order types; and region information necessary for DDM. There is no hard-coded FOM-specific data in the MÄK RTI.

### **Is the MÄK RTI written in C++?**

Yes, the MÄK RTI is fully implemented in C++, so it can be used directly by C++ federates without going through any kind of wrapper. This choice not only allowed us to make the MÄK RTI as efficient as possible, but also allowed us to create the RTIspy<sup>®</sup> plug-in API, which allows C++ developers to customize or extend the MÄK RTI (see details below). The MÄK RTI also includes a Java “cap,” that allows Java federates to use the MÄK RTI.

### **How easy is it to switch between the MÄK RTI and other RTIs?**

As long as different RTIs implement the exact header files, as dictated by the HLA Interface Specification, it is often as easy as just swapping shared libraries (DLLs or DSOs). No recompiling or re-linking is necessary, meaning that end-users (non-developers) can choose their desired RTI implementation at federation execution time. A tool vendor can create a single version of their executable that will work with the MÄK RTI, and with any other RTI that is link-compatible. Occasionally, some up-front tuning is necessary to ensure that an executable will work well with several RTIs, due to differences in recommended tick rates, or possible inadvertent dependencies on implementation characteristics of one of the RTIs. But once an executable has been tested with both, choosing which implementation to run with is just a matter of pointing to the appropriate RTI libraries.

Specifically, we have tested the dynamic-link compatibility of the HLA 1.3 version of the MÄK RTI with DMSO's RTI NG, RTI NG Pro, the MATREX RTI, and with the pRTI 1.3 from Pitch. Applications that have been built against these RTIs can typically switch to the HLA 1.3 version of the MÄK RTI without recompiling or re-linking.

For IEEE 1516, things are not as simple, because the original IEEE 1516 C++ API did not support Dynamic-Link compatibility among RTIs. For this reason, SISO developed the Dynamic-Link Compatible (DLC) C++ API for IEEE 1516. MÄK was the first to implement the SISO Standard DLC API in our RTI, which enables us to be Dynamic-Link Compatible with other RTIs that implement this standard.

### **Has the MÄK RTI been verified as HLA compliant?**

Yes – for both HLA 1.3 and the IEEE 1516-2000 version of HLA! The HLA 1.3 version of the MÄK RTI was first verified as fully HLA compliant in November 2002. The IEEE 1516-2000 version has been verified as fully compliant since February 2006. The MÄK RTI is currently undergoing verification for IEEE 1516-2010.

Full verification for HLA 1-3 and IEEE 1516-2000 requires passing a total of more than 3000 individual tests, each of which involves up to five federates making a series of RTI service calls in a particular order. The tests are designed to ensure that an RTI complies with every statement in the HLA Interface Specification.

## **What version of HLA does the MÄK RTI support? HLA 1.3? IEEE 1516-2000? IEEE 1516-2010 (HLA Evolved)?**

All of the above! The MÄK RTI has supported HLA 1.3 from the very start. By April 2004, we had released our completed IEEE 1516-2000 implementation, with support for all services. DMSO verification of that version was achieved in February 2006.

The newest version of the IEEE 1516 standard, IEEE 1516-2010 (also known as HLA Evolved) was released in 2010. In August of that year we released MÄK RTI 4.0, which added complete support for the new standard.

## **Does HLA Evolved Cost More?**

No, if you own the MÄK RTI and your maintenance is up to date, you can get the latest version of the MÄK RTI which includes the HLA 1.3, IEEE 1516-2000, and IEEE 1516-2010 versions for no additional charge.

## **Can the HLA 1.3 and IEEE 1516 versions interoperate?**

In most cases, yes! We have attempted to retain as much wire compatibility as possible between the HLA 1.3 version of the MÄK RTI and the IEEE 1516 version. In certain cases, changes in the semantics of the services between HLA versions made this impossible. However, for most services (*including* DDM, where the API differs quite a bit between HLA 1.3 and IEEE 1516), federates built to the HLA 1.3 API should be able to communicate with federates built to either of the IEEE 1516 API without any adapters or gateways. This allows you the flexibility to upgrade simulations to IEEE 1516 one at a time, rather than having to upgrade them all at once.

## **Does the MÄK RTI include an RTI Executive or other central server?**

There are many aspects of an RTI's functionality that require some form of centralized knowledge. For example, someone needs to insure that federate and object identifiers are unique, someone needs to decide when a federation is synchronized, and someone needs to keep track of which federate is the laggard in time-management calculations. There are basically two strategies that RTI developers can take in managing this centralized knowledge: Make one federate's Local RTI Component (LRC) responsible, or require a central server application commonly known as an RTI Executive or *rtiexec*. The MÄK RTI chooses the *rtiexec* approach because it is simpler (no need to transfer central-knowledge to another federate when the designated federate resigns), and usually more robust (*rtiexec* crashes are usually less frequent than federate crashes). When using the set of services that require centralized knowledge, it is necessary to run the *rtiexec* before any federates try to create or join a federation.

It is important to realize that even when you *are* using the *rtiexec*, most of your data still goes "peer-to-peer" between the federates. Specifically, attribute updates and interactions are typically sent over UDP multicast, without the *rtiexec* even getting involved. Reliable traffic goes through an RTI Forwarder, which may or may not live in the *rtiexec*, depending on configuration.

## **What is Lightweight Mode?**

Although some RTI services do require centralized knowledge, we have found that a significant fraction of HLA federations do not rely on those services. For these federations, the MÄK RTI provides a Lightweight Mode, where no *rtiexec* is necessary. In this connectionless, fully-distributed mode, all network messages are communicated directly between the federates' Local RTI Components (LRCs). A configuration setting indicates to a federate's LRC that it is running in Lightweight Mode, and should not attempt to connect to an *rtiexec*.

Lightweight Mode is well-suited for many real-time federations that do not use Time-Management, DDM, MOM, reliable transport (TCP), or synchronization points. It is particularly helpful during the development and debugging stage of a federation, when federates are unstable, and reinitializing the *rtiexec* between runs would be a hassle.

Obviously, the MÄK RTI is not fully-compliant when configured for Lightweight Mode, since not all services can be supported.

### **Does the MÄK RTI support operation over the internet or other WANs?**

Yes. If you are using reliable transport, no special configuration is necessary to run an HLA federation over the internet or other WAN, as long as there is a network route from each federate to the RTI Forwarder. When using best-effort transport, however, the default configuration is not appropriate, since UDP multicast packets typically cannot be sent over a WAN. Instead, you can take advantage of Distributed Forwarding to run an RTI Forwarder on each side of the WAN. With this configuration, each forwarder can listen for local UDP multicast packets and send them via TCP across the WAN to one or more remote RTI Forwarders.

### **How can I take advantage of Distributed Forwarding to improve performance over a WAN?**

In order to make the most efficient use of WAN bandwidth, you will want to use multiple, distributed copies of the MÄK RTI's RTI Forwarder. When you run an RTI Forwarder on each LAN, you save bandwidth by minimizing the number of copies of each packet that needs to be sent over the WAN. Instead of sending a separate copy of each packet to each remote federate, only one copy of each packet traverses each WAN link, and the remote RTI Forwarder sends the message on to each recipient on its LAN. The RTI Forwarders support sender-side filtering, so that only interested federates' LRCs will receive each message, and messages are not sent over WAN links at all when no one on the other side needs the data. Data received locally by an RTI Forwarder via UDP multicast is automatically sent to remote RTI Forwarders (if needed) via TCP.

### **Can the MÄK RTI be run behind a firewall?**

Yes. We realize that many of today's HLA federations span wide-area networks, and often each LAN is protected by a firewall. Many of our customers use the MÄK RTI in this environment. The MÄK RTI's RTI Forwarder running on each LAN serves as a single point of entry for all RTI data coming to each LAN. This makes it very easy to configure your firewall to allow HLA traffic to enter, without having to update it when federates move from machine to machine. If you use a Virtual Network scheme such as VPN, even this minimal firewall configuration becomes unnecessary.

### **What kind of fault tolerance does the MÄK RTI have?**

First, we can automatically recover from temporarily broken TCP connections, without the federate even knowing that we have disconnected and reconnected. Second, if a federate crashes or permanently loses its connection, the RTI and the rest of the federation can continue gracefully. The lost federate is removed from any Time Management or Synchronization Point calculations so that the other federates are not held up. Orphaned objects can be cleaned up and deleted.

### **How can one RTI be easier to use than another? Isn't there a standard interface specification?**

Yes, the HLA API is a standard. So from a coding perspective, integrating with one RTI is about as straightforward as integrating with another. But there are a lot of differences between various RTI implementations, in terms of how easy they are to install, and more importantly, how easy they are to configure and run.

Historically, most RTIs have required that you hand-edit a configuration file, just to do something as simple as moving the rtiexec from one machine to another, or to make sure that two different exercises on your network didn't conflict. The MÄK RTI makes this simple. It is no longer necessary to edit configuration files to configure the most commonly used options: You can switch between running with an rtiexec and Lightweight Mode, choose from a list of available rtiexecs running on your network, and even launch and configure a new rtiexec, all from a GUI that comes up automatically the first time you run a federate. In addition to choosing among pre-configured RTI Connections, you can add new Connections

by entering a port number and multicast address. You can also "force full compliance" from the GUI to ensure that all of the RTI services are enabled. If you do not want to be prompted to select an RTI Connection next time you run a federate, check the "Always attempt to use this connection" box to make a particular connection the default.

For convenience, the MÄK RTI provides an icon on the Windows Tray. From here, you can change the default RTI Connection, force resigns of local or remote federates, launch the web-based RTIspy GUI, run latency tests, enable logging, and view the RTI notification history. It also allows you to bring up the Federations View and Network Components View, which will show you the current state or network configuration of any federation on your network.

### **What is the RTIspy plug-in API?**

The RTIspy plug-in API is a C++ interface which lets you alter, extend, or query the RTI's functionality programmatically from a dynamic library you create. So, if you want to use special radio communications instead of our standard TCP/IP infrastructure, you can change it! Or, if you just want to collect detailed information about what is going on in your federation, that is possible too.

### **Can I really customize the MÄK RTI?**

Yes! The RTIspy plug-in API is a feature that is truly unique to the MÄK RTI. It allows you to build plug-ins to alter, extend, and query any aspect of the MÄK RTI's functionality.

Through the API, you can change the RTI's "wire format" by implementing new network messages, perform encryption or compression on network messages, register callbacks to be executed whenever certain services are invoked, tune performance based on program-specific requirements, and override default implementations of key services. The RTIspy API lets you tailor a proven RTI to meet your program's needs.

In part to demonstrate just some of the power of the RTIspy plug-in API, we chose to implement the RTIspy Diagnostic GUI entirely as a plug-in, using the RTIspy plug-in API.

### **What is the RTIspy GUI?**

The RTIspy GUI gathers diagnostic information directly from within the RTI, and makes it accessible through a web-based graphical user interface. Simply point your web browser at the URL associated with any Local RTI Component (LRC) or the rtiexec to find out what's going on inside the RTI. The idea of monitoring and controlling a piece of network infrastructure from a web-browser should be familiar to anyone who has used this method to set up their wireless router.

The fact that the RTI's diagnostic GUI is web-based offers many advantages. One is remote access. You don't need to be on the same machine as a federate in order to get information about its LRC's internal state. Two related benefits are centralized and distributed access. I.e., any user on the network can access information about *all* of the LRCs in the federation from one place. Conversely, several users can be browsing the same LRC at the same time. Another advantage is the ability to hide or launch the diagnostic GUI during run-time, simply by exiting and re-launching your web browser. Bookmark the URLs associated with federates of interest to quickly jump to the relevant data.

### **What diagnostic information is available through the RTIspy GUI?**

The diagnostic GUI visually displays information gleaned directly from each LRC and the rtiexec – a much more intimate view than can be provided by MOM-based tools. You can easily see the world from the perspective of the LRC – scanning the list of known objects, the interactions it has sent and received, and the current state of all FOM subscriptions and publications. For example, the RTIspy GUI makes it easy to determine whether a federate is not seeing an object because it is not subscribed to an appropriate class, because the object has never been registered, or because the federate failed to properly handle the object discovery callback. The tool also helps to solve tricky timing problems by keeping a log of all federate-invoked and RTI-invoked services, and by displaying Time-Management parameters like current logical time, requested time, and LBTS for each federate. The network-monitoring panel allows you to examine

bandwidth usage by object, both local and remote. The display also provides information on what percentage of CPU time is being used by the RTI and by FederateAmbassador callbacks. The service instrumentation panel shows exactly how many microseconds each RTI call takes. A graphical representation of your federation's topology helps you understand how your federates are distributed among various LANs, and how the RTI uses RTI Forwarders to efficiently route data across a WAN.

**How responsive is the web-based RTIspy GUI? Does it hurt RTI performance?** The RTIspy GUI uses powerful Asynchronous JavaScript (AJAX) techniques to combine the remote access and ease-of-use of a web-based interface with the responsiveness and interactivity typically associated with a native GUI application. A lightweight web server runs as a separate process outside of the RTI components, to minimize impact on the performance of the RTI itself. And of course, the actual display of RTI diagnostic information takes place in the web-browser, which can be running on a separate machine entirely.

### **Is the MÄK RTI multi-thread capable?**

Yes. A configuration parameter dictates whether the job of reading and writing packets to the network should be offloaded onto a separate asynchronous thread. Doing so has several advantages: First it means that you are less likely to drop packets due to not ticking the RTI often enough. The asynchronous thread is always reading and queuing packets as fast as possible, whereas without it, the RTI must wait for the federate to give it the necessary processor time. The other advantage is that when there is a lot of data to be read from the network, your federate is not hung up waiting for the read operation to finish. You can move on to other simulation tasks while the network-reading thread listens for incoming data. Obviously, the advantages of running in multi-threaded mode are more pronounced on a multi-core or multi-processor machine.

### **Can the MÄK RTI deliver callbacks asynchronously?**

Yes. If you enable asynchronous callbacks in the configuration file, FederateAmbassador services will be called from an asynchronous thread, without the federate having to call tick(). Of course, this requires that your federate be written in a thread-safe manner, so that RTI callbacks and the main federate thread do not access the same piece of data concurrently.

### **Can the MÄK RTI communicate via shared-memory?**

Yes. Federates that are running on the same machine can use a built-in shared memory interface to communicate. The first federate (or rtiexec) on each machine that is configured to use shared memory, automatically spawns a shared memory manager application, that maintains the message queues necessary to achieve inter-process communication. Mixing shared memory and networking in the same federation is also supported. The shared-memory manager application serves as the interface between local federates that communicate amongst themselves using shared memory, and remote federates that must be reached via TCP or UDP networking.

### **Does the MÄK RTI use CORBA?**

No. By directly using TCP and UDP sockets, we were able to ensure that data passes through as few layers of software as possible. And by designing a "wire format" specifically for the purposes of the MÄK RTI, we were able to ensure that our packets were as compact as possible.

### **How is reliable transport implemented?**

Reliable transport in the MÄK RTI is implemented on top of TCP. Because the TCP protocol supports only one recipient at a time, RTI developers need to choose between two strategies when packets need to be sent over TCP to multiple recipients. Either each federate's LRC needs a connection to every other federate, or some forwarding scheme must be used, where a packet is sent to a forwarder, who sends a copy to each recipient. The MÄK RTI implements a forwarder strategy. (The RTI Forwarder will be automatically started by the rtiexec if it is not already running.) This approach is simpler, because it means each federate needs only one TCP connection, and more efficient in CPU time for federates, because federates can move on to other simulation responsibilities after sending a single copy of each packet. The

MÄK RTI's RTI Forwarder employs "smart" forwarding – performing sender-side filtering to ensure that packets are only sent to those federates that need them.

### **Can I really get the RTI for free?**

Yes, with limitations! You can download the MÄK RTI from <http://www.mak.com/products/rti.php>, and start using it immediately in unlicensed mode, without requiring a license key. In unlicensed mode, all RTI services are available, but the RTI only allows two federates to join each federation execution. Unlicensed mode is useful for developers who need to run and test their federate (along with one other federate) without consuming one of their lab's RTI licenses. It is useful for people who want to take their federate on the road for demonstrations, without requiring an RTI license on their laptop. It is useful for developers who sometimes work at home, and cannot justify purchasing an extra RTI license for remote development. And, of course, it is useful for evaluating and becoming familiar with the MÄK RTI before purchasing.

When you want to upgrade to a full license, in order to run in "real" federations with more than two federates, all you need is a license key. No new RTI download is necessary.

### **How is the MÄK RTI being sold?**

The MÄK RTI has one configuration. It is fully HLA compliant, including support for all RTI services, both normal and Lightweight Mode operation, and there is no limit on the number of federates per federation. The RTI Standard license also includes web-based RTIspy Diagnostic GUI, the RTIspy plug-in API, and distributed RTI Forwarding for more efficient WAN operation. RTI Standard licenses are platform independent and sold on a per-federate basis.

### **What is the RTI Program Protection Plan?**

We recognize that large programs have special RTI needs. Through our RTI Program Protection Plan, we can put together a custom package of licenses, support, and consulting tailored for your program. In addition to a site license good for an unlimited number of federates, a typical plan might include a dedicated support engineer, on-demand porting to new compiler versions or operating systems, on-site consulting, or custom features.

The RTI Program Protection Plan allows program managers to lock in a price up front, so that they can budget for the future, without fear of unexpected costs creeping in later.

### **Is source code available?**

Source code is not included with releases of the MÄK RTI. However, we are certainly willing to consider special arrangements that would include licensing of our RTI source code. That being said, we have tried to address in other ways the three main underlying reasons that customers sometimes believe they need source code:

- 1) Ability to extend: The RTIspy API allows customers to extend the MÄK RTI, add features, or modify behavior. By providing a well-documented API (header files and libraries), you can add your extensions more easily than if you had to wade through hundreds of thousands of lines of source code.
- 2) Ability to port: MÄK has demonstrated a willingness to port to just about any platform for which there is demand. Often, we must charge an additional porting fee to build on a non-core platform, but it will usually be much cheaper than porting our source code yourself, especially when you take into account the recurring cost of merging your changes in each time a new release comes out.
- 3) Peace of mind: Our technical support team stands ready to quickly fix bugs and if necessary, release patches, so that you will never have to "just wait for the next feature release" in order to have critical problems solved. You will not need to modify source code to get problems fixed in a timely manner.

### **What platforms does the MÄK RTI support?**

The full MÄK RTI distribution, including fully-compliant support for all services, the rtiexec, the RTIspy API, RTI Forwarders, and the web-based RTIspy diagnostic GUI, is supported on both Windows and Linux (several different compiler versions on each).

We are also willing to port the MÄK RTI to just about any platform you *want* us to support. We have done demand-based porting of various releases to DEC Alpha, PC-Solaris, HP, AIX, PowerUX, VxWorks, and OpenVMS. Let us know what you need.

### **What about technical support and upgrades?**

At MÄK, technical support is not just an afterthought. Our reputation for supporting our customers is one of the key reasons that people choose our products. When you call or email us with questions, you speak directly to our product developers who know the software inside and out. When you buy MÄK's products, you can be sure that MÄK will be in your corner as you work towards successful completion of your HLA project.

If you need assistance that goes beyond the scope of technical support, our engineering services group is available to do customization, federation-specific performance tuning, or consulting on demand. We have had many MÄK RTI customers buy our engineers' time to help them use it to best advantage.

With maintenance, you are entitled to upgrades when they are released. We are constantly working to improve our products, and the fact that our developers provide customer support directly means that we are able to quickly react to customer feedback and integrate desired features in a timely manner.

### **What are the future plans for the MÄK RTI?**

Although the MÄK RTI provides a robust and efficient infrastructure for HLA federations today, there is still a lot of work we plan to do to extend and enhance its capabilities.

Meanwhile, we continue to work to make the MÄK RTI even easier to use and configure. Connecting your federate to an HLA network should be as easy as connecting your laptop to a WiFi network, and we are working hard to reach that goal. In version 3.3 of the MÄK RTI, we added a new GUI-based mechanism called the RTI Assistant for setting up and choosing your RTI network connection parameters. So it is no longer necessary to edit a configuration file to set the most commonly used options. But going forward, we plan to expand that capability, so that even advanced configuration can be done through preferences dialogs.

The RTI Assistant does more than just allow you to configure your RTI settings, however. It also allows you to monitor basic federation state information, view the network configuration for your federation, examine log files at run time to diagnose potential issues, and receive important notifications about warnings or errors. We also plan to expand this GUI to provide even more diagnostic information, so that at any time in any size federation you can see what classes a federate has subscribed to, what objects it has published, and how much traffic it is generating.

Our intention is for the MÄK RTI to be the obvious choice for HLA users for years to come.