QualNet 7.1
Federation Interfaces Library

August 2013

SCALABLE
NETWORK TECHNOLOGIES

Table of Contents

1 Introduction

Multiple simulators can be used to simulate different aspects of the same scenario. The results of such a co-operative simulation can be more realistic and meaningful than those obtained by using any single simulator. When multiple simulation systems are integrated together into a coordinated "synchronized" environment, they create a "federation". An example of this is an enhanced air traffic control training system that is a created as a federation of tools that predict weather patterns and training platforms to teach air traffic control operators. This system realistically models the effects on ground-to-air communications due to variable weather behavior.

In a federated environment, the multiple simulators pass various types of messages among them to coordinate their actions and responses. One system might model entity movement over terrain. Distance and speed updates from this system are passed to another simulator in the federation to enable real-time recalculation of other effects, such as radio communication between entities.

The key to creating an effective federation of simulation systems is that they all need to interoperate using a common communications protocol. The modeling and simulation industry has standardized a small number of protocols in order to simplify this process. SCALABLE supports three of the most popular standard interfaces for enabling federation: High Level Architecture (HLA), Distributed Interactive Simulation (DIS) and IP sockets.

VR-Link Interface

The VR-Link networking toolkit from MAK Technologies (www.mak.com) provides a protocol-independent API that abstracts away the details of simulation networking protocols. The QualNet VR-Link interface allows you to network simulators and virtual reality applications using HLA or DIS.

See Chapter 2 for details of the QualNet Socket Interface.

High Level Architecture

High Level Architecture (HLA) is a general-purpose architecture for distributed computer simulation systems. Computer simulations can interact (i.e., communicate data and synchronize actions) with other computer simulations regardless of the computing platforms. The interaction between simulations is managed by a Run-Time Infrastructure (RTI). HLA is an interoperability standard for distributed simulation used to support analysis, engineering and training in a number of different domains.

In HLA terminology, the collection of communicating simulations is called a *federation* and each simulation is called a *federate.* The object and interaction classes used in the federation are defined in a module called Federation Object Model (FOM). Information is exchanged between simulations using this FOM.

See Section 2.2 for details of the QualNet HLA interface.

Distributed Interactive Simulation
Distributed Interactive Simulation (DIS) is an IEEE standard for interfacing multiple simulation tools into a single, real-time simulation. The transport of information between simulators is performed using UDP and broadcast and/or multicast IP. Although superseded by HLA and IEEE 1516, DIS still remains popular for its simplicity of operation and the ease of creating a DIS interface.

See Section 2.3 for details of the QualNet DIS interface.

Socket Interface
The Socket Interface provides inter-process communication between QualNet and an external program over a TCP socket, with QualNet acting as the server and the external program as the client. Several types of messages can be sent between the two processes.

See Chapter 3 for details of the QualNet Socket Interface.

....

1.1 Conventions Used

1.1.1 Format for Command Line Configuration

This section describes the general format for specifying parameters in input files, the precedence rules for parameters, and the conventions used in the description of command line configuration for each model.

1.1.1.1 General Format of Parameter Declaration

The general format for specifying a parameter in an input file is:

```
[<Qualifier>] <Parameter Name> [<Index>] <Parameter Value>
```

where

`<Qualifier>`	The qualifier is optional and defines the scope of the parameter declaration. The scope can be one of the following: Global, Node, Subnet, and Interface. Multiple instances of a parameter with different qualifiers can be included in an input file. Precedence rules (see Section 1.1.1.2) determine the parameter value for a node or interface.

Global: The parameter declaration is applicable to the entire scenario (to all nodes and interfaces), subject to precedence rules. The scope of a parameter declaration is global if the qualifier is not included in the declaration.

Example:

```
MAC-PROTOCOL              MACDOT11
```

Node: The parameter declaration is applicable to specified nodes, subject to precedence rules. The qualifier for a node-level declaration is a list of space-separated node IDs or a range of node IDs (specified by using the keyword `thru`) enclosed in square brackets.

Example:

```
[5 thru 10] MAC-PROTOCOL              MACDOT11
```

Subnet: The parameter declaration is applicable to all interfaces in specified subnets, subject to precedence rules. The qualifier for a subnet-level declaration is a space-separated list of subnet addresses enclosed in square brackets. A subnet address can be specified in the IP dot notation or in the QualNet N syntax.

Example:

```
[N8-1.0 N2-1.0] MAC-PROTOCOL        MACDOT11
```

Interface: The parameter declaration is applicable to specified interfaces. The qualifier for an interface-level declaration is a space-separated list of subnet addresses enclosed in square brackets.

Example:

```
[192.168.2.1 192.168.2.4] MAC-PROTOCOL MACDOT11
```

`<Parameter Name>`	Name of the parameter.
`<Index>`	Instance of the parameter to which this parameter declaration is applicable, enclosed in square brackets. This should be in the range 0 to n -1, where n is the number of instances of the parameter.
	The instance specification is optional in a parameter declaration. If an instance is not included, then the parameter declaration is applicable to all instances of the parameter, unless otherwise specified.
`<Parameter Value>`	Value of the parameter.

Note: There should not be any spaces between the parameter name and the index.

Examples of parameter declarations in input files are:

```
PHY-MODEL                                        PHY802.11b
[1] PHY-MODEL                                    PHY802.11a
[N8-1.0] PHY-RX-MODEL                            BER-BASED
[8 thru 10] ROUTING-PROTOCOL                     RIP
[192.168.2.1 192.168.2.4] MAC-PROTOCOL           GENERICMAC
NODE-POSITION-FILE                               ./default.nodes
PROPAGATION-CHANNEL-FREQUENCY[0]                 2.4e9
[1 2] QUEUE-WEIGHT[1]                            0.3
```

Note In the rest of this document, we will not use the qualifier or the index in a parameter's description. Users should use a qualifier and/or index to restrict the scope of a parameter, as appropriate.

1.1.1.2 Precedence Rules

Parameters without Instances

If the parameter declarations do not include instances, then the following rules of precedence apply when determining the parameter values for specific nodes and interfaces:

Interface > Subnet > Node > Global

This can be interpreted as follows:

- The value specified for an interface takes precedence over the value specified for a subnet, if any.
- The value specified for a subnet takes precedence over the value specified for a node, if any.
- The value specified for a node takes precedence over the value specified for the scenario (global value), if any.

Parameters with Instances

If the parameter declarations are a combination of declarations with and without instances, then the following precedence rules apply (unless otherwise stated):

Interface[i] > Subnet[i] > Node[i] > Global[i] > Interface > Subnet > Node > Global

This can be interpreted as follows:

- Values specified for a specific instance (at the interface, subnet, node, or global level) take precedence over values specified without the instance.

- For values specified for the same instance at different levels, the following precedence rules apply:
    - The value specified for an interface takes precedence over the value specified for a subnet, if any, if both declarations are for the same instance.
    - The value specified for a subnet takes precedence over the value specified for a node, if any, if both declarations are for the same instance.
    - The value specified for a node takes precedence over the value specified for the scenario (global value), if any, if both declarations are for the same instance.

1.1.1.3 Parameter Description Format

In the Model Library, most parameters are described using a tabular format described below. The parameter description tables have three columns labeled "Parameter", "Values", and "Description". Table 1-1 shows the format of parameter tables. Table 1-3 shows examples of parameter descriptions in this format.

TABLE 1-1.   Parameter Table Format

Parameter	Values	Description
<Parameter Name>	*<Type>*	*<Description>*
<Designation>	[*<Range>*]	
<Scope>	[*<Default Value>*]	
[*<Instances>*]	[*<Unit>*]	

Parameter Column

The first column contains the following entries:

- ***<Parameter Name>*:** The first entry is the parameter name (this is the exact name of the parameter to be used in the input files).

- ***<Designation>:*** This entry can be *Optional* or *Required*. These terms are explained below.
    - ***Optional***: This indicates that the parameter is optional and may be omitted from the configuration file. (If applicable, the default value for this parameter is included in the second column.)
    - ***Required***: This indicates that the parameter is mandatory and must be included in the configuration file.

- ***<Scope>:*** This entry specifies the possible scope of the parameter, i.e., if the parameter can be specified at the global, node, subnet, or interface levels. Any combination of these levels is possible.If the parameter can be specified at all four levels, the keyword "All" is used to indicate that.

    Examples of scope specification are:
    Scope: All
    Scope: Subnet, Interface
    Scope: Global, Node

- ***<Instances>:*** If the parameter can have multiple instances, this entry indicates the type of index. If the parameter can not have multiple instances, then this entry is omitted.

Examples of instance specification are:

Instances: channel number

Instances: interface index

Instances: queue index

Values Column

The second column contains the following information:

- *<Type>*: The first entry is the parameter type and can be one of the following: Integer, Real, String, Time, Filename, IP Address, Coordinates, Node-list, or List. If the type is a List, then all possible values in the list are enumerated below the word "List". (In some cases, the values are listed in a separate table and a reference to that table is included in place of the enumeration.)

  Table 1-2 shows the values a parameter can take for each type.

TABLE 1-2.   Parameter Types

Type	Description
Integer	Integer value Examples: `2, 10`
Real	Real value Examples: `15.0, -23.5, 2.0e9`
String	String value Examples: `TEST, SWITCH1`
Time	Time value expressed in QualNet time syntax (refer to *QualNet User's Guide*) Examples: `1.5S, 200MS, 10US`

Filename	Name of a file in QualNet filename syntax (refer to *QualNet User's Guide*)
	 Examples:
	 `../../data/terrain/los-angeles-w`
	 (For Windows and UNIX)
	 `C:\scalable\qualnet\7.1\scenarios\WF\WF.nodes`
	 (For Windows)
	 `/root/scalable/qualnet/7.1/scenarios/WF/WF.nodes`
	 (For UNIX)

Path	Path to a directory in QualNet path syntax (refer to *QualNet User's Guide*)
	 Examples:
	 `../../data/terrain` (For Windows and UNIX)
	 `C:\scalable\qualnet\7.1\scenarios\default`
	 (For Windows)
	 `/root/scalable/qualnet/7.1/scenarios/default`
	 (For UNIX)

IP Address	IPv4 or IPv6 address Examples: `192.168.2.1, 2000:0:0:0::1`

TABLE 1-2.   Parameter Types (Continued)

Type	Description
IPv4 Address	IPv4 address Examples: `192.168.2.1`
IPv6 Address	IPv6 address Examples: `2000:0:0:0::1`
Coordinates	Coordinates in Cartesian or Lat-Lon-Alt system. The altitude is optional. Examples: `(100, 200, 2.5)`, `(-25.3478, 25.28976)`
Node-list	List of node IDs separated by commas and enclosed in "{" and "}". Examples: `{2, 5, 10}`, `{1, 3 thru 6}`
List	One of the enumerated values. Example: See the parameter `MOBILITY` in Table 1-3.

Note: If the parameter type is List, then options for the parameter available in QualNet and the commonly used model libraries are enumerated. Additional options for the parameter may be available if some other model libraries or addons are installed. These additional options are not listed in this document but are described in the corresponding model library or addon documentation.

- ***<Range>:*** This is an optional entry and is used if the range of values that a parameter can take is restricted. The permissible range is listed after the label "*Range:*" The range can be specified by giving the minimum value, the maximum value, or both. If the range of values is not restricted, then this entry is omitted.

  If both the minimum and maximum values are specified, then the following convention is used to indicate whether the minimum and maximum values are included in the range:

`(min, max)`	`min` < parameter value < `max`
`[min, max)`	`min` ≤ parameter value < `max`
`(min, max]`	`min` < parameter value ≤ `max`
`[min, max]`	`min` ≤ parameter value ≤ `max`

`min` (or `max`) can be a parameter name, in which case it denotes the value of that parameter.

Examples of range specification are:

Range: ≥ `0`
Range: `(0.0, 1.0]`
Range: `[1, MAX-COUNT]`
Range: `[1S, 200S]`

Note: If an upper limit is not specified in the range, then the maximum value that the parameter can take is the largest value of the type (integer, real, time) that can be stored in the system.

- **<Default>:** This is an optional entry which specifies the default value of an optional or conditional-optional parameter. The default value is listed after the label "*Default:*"

- **<Unit>:** This is an optional entry which specifies the unit for the parameter, if applicable. The unit is listed after the label "*Unit:*". Examples of units are: meters, dBm, slots.

Description Column

The third column contains a description of the parameter. The significance of different parameter values is explained here, where applicable. In some cases, references to notes, other tables, sections in the User's Guide, or to other model libraries may be included here.

Table 1-3 shows examples of parameter descriptions using the format described above.

TABLE 1-3.   Example Parameter Table

Parameter	Values	Description
MOBILITY *Optional* *Scope:* Global, Node	List: - NONE - FILE - GROUP-MOBILITY - RANDOM-WAYPOINT *Default:* NONE	Mobility model used for the node. If MOBILITY is set to NONE, then the nodes remain fixed in one place for the duration of the simulation. See Table 7-11 for a description of mobility models.
BACKOFF-LIMIT *Required* *Scope:* Subnet, Interface	Integer *Range:* [4,10) *Unit:* slots	Upper limit of backoff interval after collision. A backoff interval is randomly chosen between 1 and this number following a collision.
IP-QUEUE-PRIORITY-QUEUE-SIZE *Required* *Scope:* All *Instances:* queue index	Integer *Range:* [1, 65535] *Unit:* bytes	Size of the output priority queue.
MAC-DOT11-DIRECTIONAL-ANTENNA-MODE *Optional* *Scope:* All	List - YES - NO *Default:* NO	Indicates whether the radio is to use a directional antenna for transmission and reception.

1.1.2 Format for GUI Configuration

The GUI configuration section for a model outlines the steps to configure the model using the GUI. The following conventions are used in the GUI configuration sections:

Path to a Parameter Group

As a shorthand, the location of a parameter group in a properties editor is represented as a path consisting of the name of the properties editor, name of the tab within the properties editor, name of the parameter group within the tab (if applicable), name of the parameter sub-group (if applicable), and so on.

Example

The following statement:

Go to **Default Device Properties Editor > Interfaces > Interface # > MAC Layer**

is equivalent to the following sequence of steps:

1. Open the Default Device Properties Editor for the node.

2. Click the **Interfaces** tab.

3. Expand the applicable Interface group.

4. Click the **MAC Layer** parameter group.

The above path is shown in Figure 1-1.
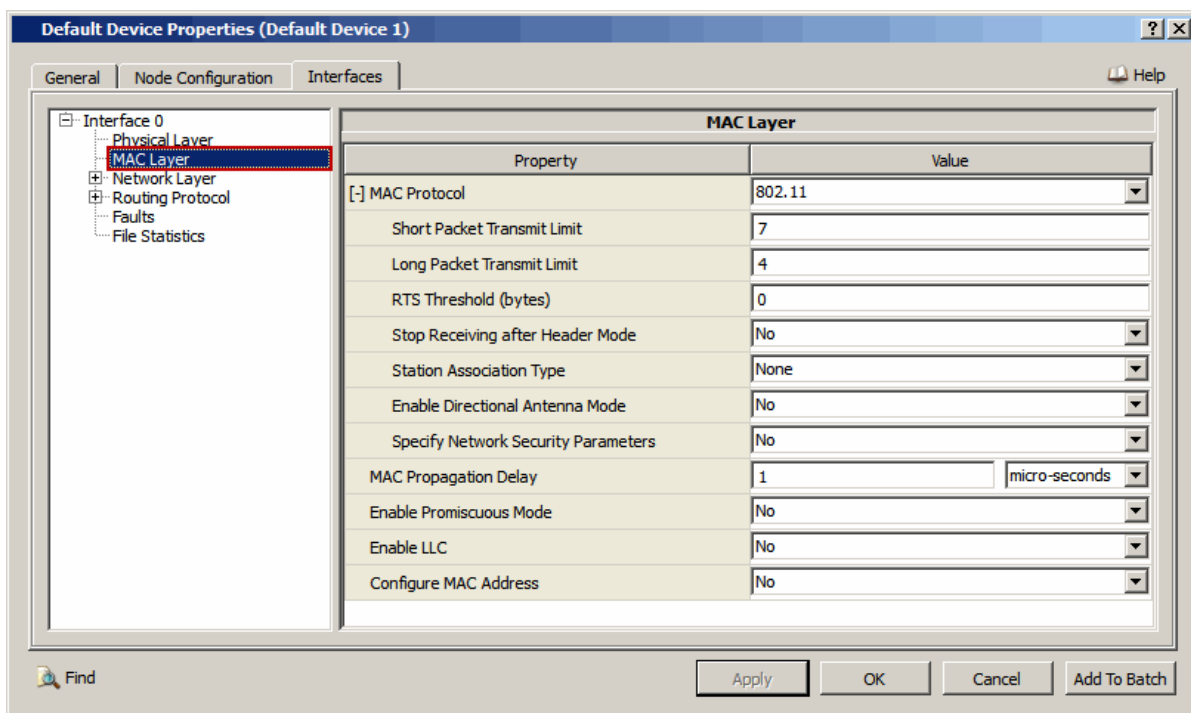
FIGURE 1-1.   Path to a Parameter Group

Path to a Specific Parameter

As a shorthand, the location of a specific parameter within a parameter group is represented as a path consisting of all ancestor parameters and their corresponding values starting from the top-level parameter. The value of an ancestor parameter is enclosed in square brackets after the parameter name.

Example

The following statement:

Set **MAC Protocol** *[= 802.11]* **> Station Association Type** *[= Dynamic]* **> Set Access Point** *[= Yes]* **>** *Enable Power Save Mode* to *Yes*

is equivalent to the following sequence of steps:

1. Set **MAC Protocol** to *802.11.*

2. Set **Station Association Type** to *Dynamic.*

3. Set **Set Access Point** to *Yes.*

4. Set **Enable Power Save Mode** to *Yes.*

The above path is shown in Figure 1-2.
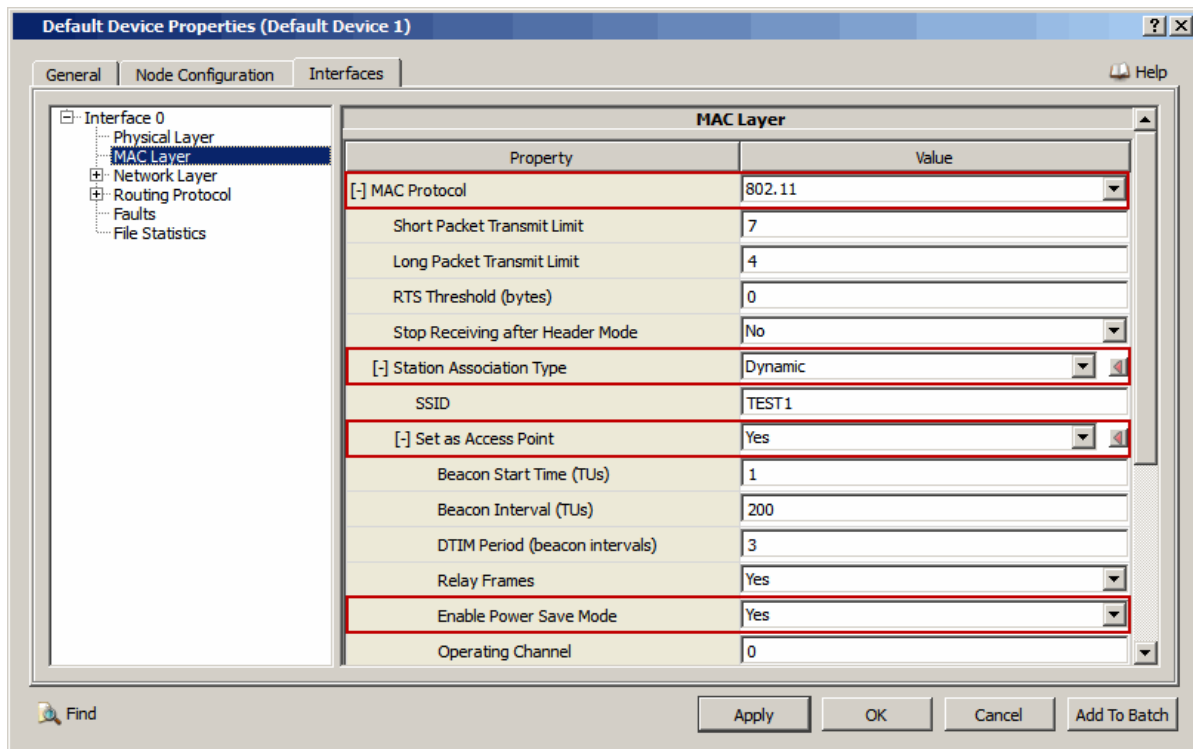
FIGURE 1-2.   Path to a Specific Parameter

Parameter Table

GUI configuration of a model is described as a series of a steps. Each step describes how to configure one or more parameters. Since the GUI display name of a parameter may be different from the name in the configuration file, each step also includes a table that shows the mapping between the GUI names and command line names of parameters configured in that step. For a description of a GUI parameter, see the description of the equivalent command line parameter in the command line configuration section.

The format of a parameter mapping table is shown in Table 1-4.

TABLE 1-4.   Mapping Table

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
<GUI Display Name>	*<Scope>*	*<Command Line Parameter Name>*

The first column, labeled "GUI Parameter", lists the name of the parameter as it is displayed in the GUI.

The second column, labeled "Scope of GUI Parameter", lists the level(s) at which the parameter can be configured. *<Scope>* can be any combination of: Global, Node, Subnet, Wired Subnet, Wireless Subnet, Point-to-point Link, and Interface.

Table 1-5 lists the Properties Editors where parameters with different scopes can be set.

Notes: 1. Unless otherwise stated, the "Subnet" scope refers to "Wireless Subnet".

       2. The scope column can also refer to Properties Editors for special devices and network components (such as ATM Device Properties Editor) which are not included in Table 1-5.

TABLE 1-5.   Properties Editors for Different Scopes

Scope of GUI Parameter	Properties Editor
Global	Scenario Properties Editor
Node	Default Device Properties Editor (General and Node Configuration tabs)
Subnet Wireless Subnet	Wireless Subnet Properties Editor
Wired Subnet	Wired Subnet Properties Editor
Point-to-point Link	Point-to-point Link Properties Editor
Interface	Interface Properties Editor, Default Device Properties Editor (Interfaces tab)

The third column, labeled "Command Line Parameter", lists the equivalent command line parameter.

Note: For some parameters, the scope may be different in command line and GUI configurations (a parameter may be configurable at fewer levels in the GUI than in the command line).

Table 1-6 is an example of a parameter mapping table.

TABLE 1-6.   Example Mapping Table

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Define Area	Node	OSPFv2-DEFINE-AREA
OSPFv2 Configuration File	Node	OSPFv2-CONFIG-FILE
Specify Autonomous System	Node	N/A
Configure as Autonomous System Boundary Router	Node	AS-BOUNDARY-ROUTER
Inject External Route	Node	N/A
Enable Stagger Start	Node	OSPFv2-STAGGER-START

2 VR-Link Interface

VR-Link is a protocol-independent API that abstracts away the details of simulation networking protocols. VR-Link allows you to network simulators and virtual reality applications using the industry standard High Level Architecture (HLA) or the Distributed Interactive Simulation (DIS) protocol.

Section 2.1 describes the systems requirements and instructions for installing and compiling the QualNet Federation Interfaces Library.

Section 2.2 describes how to use the HLA protocol.

Section 2.3 describes how to use the DIS protocol.

2.1 System Requirements, Installation, and Compilation

2.1.1 System Requirements

Refer to *QualNet Installation Guide* and *QualNet Distributed Reference Guide* for systems requirements for running QualNet. In addition, to use VR-Link, you also need:

- QualNet Federation Interfaces Library: See Section 2.1.2 for instructions for installing the Federation Interfaces Library.
- An RTI installation that follows the HLA 1.3 or HLA 1516 standard (needed only for using the HLA protocols).

Note: We strongly recommend MAK RTI (version 2.1 or higher). Not only is it one of the most popular RTIs on the market, it also the one most tested for use with VR-Link.

MAK RTI 4.1 can be downloaded using the following links:

- For Windows platforms using VC 9:
  ftp://ftp.mak.com/out/patches/makRti4.1.1f-win32-win64-msvc++9.0-setup-nwpx.exe
- For Windows platforms using VC 10:
  ftp://ftp.mak.com/out/patches/makRti4.1.1f-win32-win64-msvc++10.0-setup-vnrf.exe
- For the supported Linux platforms (Centos 5.9, Red Hat Linux Enterprise 5.9, and Ubuntu 12.04 LTS):
  ftp://ftp.mak.com/out/patches/makRti4.1.1f-redhatentws5-gcc4.1.2-dybk.tar.gz

Additional Requirements for Ubuntu Platforms

- Ubuntu (32-bit and 64-bit) platforms also require libz to be installed. You can install libz using the package manager with the following command:

```
sudo apt-get install libz-dev
```

- To run VR Forces or MAK RTI, Ubuntu (64-bit) platforms also require the 32-bit compatibility libraries to be installed. You can install the 32-bit compatibility libraries using the package manager with the following command:

```
sudo apt-get install ia32-libs
```

Additional Requirements for Compiling QualNet with VR-Link Interface

If you need to recompile QualNet with the VR-Link interface (without making any modifications to the HLA or DIS protocol models), refer to *QualNet Programmer's Guide* for requirements for compiling QualNet on shared memory systems and to *QualNet Distributed Reference Guide* for requirements for compiling QualNet on distributed architectures. (You must have QualNet Federation Interfaces Library and an RTI installation compliant with HLA 1.3 or HLA 1516 installed on the system.)

If you have installed an RTI other than MAK RTI, then make sure that the RTI provides the RTI C++ header files and libraries for development. You cannot recompile QualNet with the VR-Link interface (with modifications to the HLA or DIS protocol models) with a Java only RTI or if you have only installed the RTI runtime libraries. Any RTI implementation that follows the HLA 1.3 or HLA 1516 standard should work but may require some custom settings.

If you modify the HLA or DIS protocol models and want to recompile QualNet with the VR-Link interface, you also need VR-Link 4.0.5 (or higher) development installation and a VR-Link development license. For information on installing VR-Link go to f http://www.mak.com/products/vrlink.php.

2.1.2 Installation

Before installing the Federation Interfaces Library, you must install QualNet 7.1. Refer to *QualNet Installation Guide* for instructions for installing QualNet 7.1.

This section describes how to install the Federation Interfaces Library on Windows (see Section 2.1.2.1) and Linux (see Section 2.1.2.2) platforms.

2.1.2.1 Installation on Windows

To install the Federation Interfaces Library on Windows, perform the following steps:

1. Download the installation package (federation-interfaces-1.1-windows-installer.exe) from the QualNet download page or load it from the installation CD.

2. Double click on the installation package.

3. Follow the installation prompts.

2.1.2.2 Installation on Linux

The Federation Interfaces Library can be installed locally on a machine by running the installer's GUI (see Section 2.1.2.2.1). The Federation Interfaces Library can be installed on a remote machine by running the installer from the command line using SSH (see Section 2.1.2.2.2).

2.1.2.2.1 Installation Using Installer's GUI

To install the Federation Interfaces Library on a Linux system using the installer's GUI, perform the following steps:

1. Download the installation package from the QualNet download page or load it from the installation CD.

    • For 32-bit platforms, the installation package is federation-interfaces-1.1-linux-installer-32bit

    • For 64-bit platforms, the installation package is federation-interfaces-1.1-linux-installer-64bit

2. Double click on the installation package.

3. Follow the installation prompts.

2.1.2.2.2 Installation from Command Line

To install the Federation Interfaces Library on a Linux system using the installer's command line interface, perform the following steps:

1. Download the installation package from the QualNet download page or load it from the installation CD.

    • For 32-bit platforms, the installation package is federation-interfaces-1.1-linux-installer-32bit

    • For 64-bit platforms, the installation package is federation-interfaces-1.1-linux-installer-64bit

2. Open a command window and change the directory to the location where you downloaded the installation package.

3. Type the following command:

    For 32-bit platforms:

    ```
    ./federation-interfaces-1.1-linux-installer-32bit --mode text
    ```

    For 64-bit platforms:

    ```
    ./federation-interfaces-1.1-linux-installer-64bit --mode text
    ```

4. Follow the installation prompts.

2.1.3 Environment Variables for Running QualNet with HLA and DIS

This section describes the environment variables that must be set in order to run QualNet with HLA and DIS.

> **Note:** If you modify the HLA or DIS protocol models and want to recompile QualNet with VR-Link, then you must also set the environment variables described in Section 2.1.5.1.

Environment Variables for HLA on Windows
- `PATH` must include `QUALNET_HOME/bin`
- `PATH` must include `(RTI_HOME)/bin`, where `(RTI_HOME)` is the location where RTI is installed

Environment Variables for HLA on Linux
- `PATH` must include `QUALNET_HOME/bin`
- `LD_LIBRARY_PATH` must include `QUALNET_HOME/bin`
- `LD_LIBRARY_PATH` must include `(RTI_HOME)/bin`, where`(RTI_HOME)` is the location where RTI is installed

Environment Variables for DIS on Windows
- `PATH` must include `QUALNET_HOME/bin`

Environment Variables for DIS on Linux
- `PATH` must include `QUALNET_HOME/bin`
- `LD_LIBRARY_PATH` must include `QUALNET_HOME/bin`

RTI Environment Variables for HLA
If you have installed the MAK RTI and want to use HLA, then also set the following environment variables (these apply to both Windows and Linux platforms):
- `MAK_RTIDIR` must be set to the MAK RTI installation directory
- `PATH` should include `MAK_RTIDIR/bin`
- It is recommended that `RTI_RID_FILE` be set to the absolute path (including the file name) to the file rid.mtl in the MAK RTI installation directory.

If you want to use HLA and have installed an RTI other than the MAK RTI, then refer to the product documentation for that RTI for setting environment variables that need to be set.

2.1.4 Recommended Customizations for MAK RTI

If you have installed MAK RTI, then do the following to prevent pop-up warnings for disabled services in HLA 1516:

1. Open the file rid.mtl in the MAK installation directory.

2. Set `RTI_enableWarningsForDisabledServices` to `0`.

2.1.5 Recompiling QualNet with VR-Link

If you have installed the Federation Interfaces Library and have a license for it, then you can use the HLA and DIS models without recompiling QualNet.

However, you will need to recompile QualNet if:

• You have made changes to models other than HLA and DIS: In this case you only need to modify a makefile to include VR-Link before recompiling QualNet (see Section 2.1.5.1 and Section 2.1.5.2).

• You have made changes to the HLA or DIS models: In this case, you need to set additional environment variables and modify additional makefiles before recompiling QualNet. Contact the support desk at Scalable Network Technologies (support@scalable-networks.com) for details.

2.1.5.1 Compilation on Windows

To compile QualNet with VR-Link support on Windows, perform the following steps:

1. Open a command window and change the directory to QUALNET_HOME/main. (Refer to Chapter 2 of *QualNet Programmer's Guide* for commands to open the right command window on different platforms.)

2. Open the file QUALNET_HOME/main/Makefile-addons-windows with a text editor.
Change the line

```
#include ../interfaces/vrlink/Makefile-windows
```
to
```
include ../interfaces/vrlink/Makefile-windows
```

3. Copy the makefile for your compiler to Makefile. For example, if you are using VC9 on a 32-bit platform, use the following command to make a copy of the makefile:

```
copy Makefile-windows-vc9 Makefile
```

Refer to *QualNet Programmer's Guide* for the list of makefiles for different compilers.

4. Use the following commands to remove all object files and recompile:

```
nmake clean
nmake
```

2.1.5.2 Compilation on Linux

To compile QualNet with VR-Link support on Linux, perform the following steps:

1. Open a command window and change the directory to QUALNET_HOME/main.

2. Open the file QUALNET_HOME/main/Makefile-addons-unix with a text editor.
   Change the line

   ```
   #include ../interfaces/vrlink/Makefile-unix
   ```
   to
   ```
   include ../interfaces/vrlink/Makefile-unix
   ```

3. Copy the makefile for your compiler to Makefile. For example, for Red Hat Enterprise Linux 5.9 and other Linux distributions with glibc 2.5 and gcc 4.1 on a 32-bit platform, use the following command to make a copy of the makefile:

   ```
   cp Makefile-linux-glibc-2.5-gcc-4.1 Makefile
   ```

   Refer to *QualNet Programmer's Guide* for the list of makefiles for different compilers.

4. Use the following commands to remove all object files and recompile:

   ```
   make clean
   make
   ```

∙∙∙

2.2 High Level Architecture (HLA) Protocol

High Level Architecture (HLA) is a specification that enables two or more software programs (usually simulation software) to interoperate. The software programs communicate with each other through a Run-Time Infrastructure (RTI) module, which implements the HLA protocol specification. The RTI software typically comprises a RTI Executive process and a RTI library. Some RTIs have an additional process called the Federation Executive process. Typically, RTI library functions are added to each simulation and each simulation is linked with RTI libraries at compile time. The RTI Executive process is a globally known process that initializes RTI components and provides services to the distributed simulations. This enables the simulations to communicate with each other.

In HLA terminology, the collection of communicating simulations is called a *federation* and each simulation is called a *federate*. The object and interaction classes used in the federation are defined in a module called Federation Object Model (FOM). Information is exchanged between simulations using this FOM.

For example, a FOM could define an object class representing a plane, with latitude, longitude, and altitude values as object attributes. A federate creates an instance of a plane and periodically updates its position (the latitude, longitude, and altitude attributes). Through the RTI, another federate can access this plane and receive its position updates.

Interaction classes are defined to implement interactions between federates. Whereas objects and object attributes are more permanent in nature, interactions are transitory. For example, when simulating the blinking of a light on a plane, each blink can be sent as an interaction.

Federates announce their ability to create objects, update object attributes, and send interactions by publishing them. Federates subscribe to object classes, object attributes, and interaction classes of interest created by other federates.

QualNet and HLA Time Management
The HLA standard defines a number of optional time management services. However, Real-time Platform Reference Federation Object Model (RPR FOM) federate support for these services is optional and they are not supported in QualNet. Instead, QualNet operates with time stepped, clock driven, independent time advance. This clock driven simulation is considered "real-time" because each second of wall clock time is equivalent to one second in the virtual world.

Supported HLA Standards
QualNet supports the following HLA standards:

- HLA 1.3
- HLA 1516 (SISO DLC)

Using the HLA Protocol
Section 2.2.1 describes the general procedure for using the HLA protocol.

Section 2.2.2 describes how to create QualNet scenarios for use with the HLA protocol.

Section 2.2.3 describes the sequence of events when a scenario is simulated in a federation comprising QualNet and an external simulator.

Section 2.2.4 describes how to use the HLA protocol to run a simulation in testfed and QualNet. testfed (test federate) program is a command-line based program included in the QualNet distribution which

federates directly with QualNet. testfed is primarily used to test the QualNet HLA protocol without the overhead of running a full constructive simulation tool. See Appendix A for details of testfed.

Section 2.2.5 describes how to use the HLA protocol to run a simulation in VR Forces and QualNet.

Appendix D describes the Real-time Platform Reference Federation Object Model (RPR FOM) used by the QualNet HLA protocol, the RPR FOM 1.0 Extensions Interface Communications Document, and the Dynamic Statistics Interface Communications Document.

2.2.1 General Procedure for Using HLA Protocol

In general, simulating a scenario using the HLA protocol comprises the steps listed below. These steps are explained in the following sections.

1. Create a simulation scenario for the external simulator.

2. Create a QualNet scenario that is compatible with the external simulation scenario.

3. Start the RTI.

4. Start the external simulator and load the scenario.

5. Start QualNet and run the QualNet scenario equivalent to the external simulator scenario.

6. Run the simulation in the external simulator.

2.2.2 Creating Scenarios

For QualNet to provide communication effects to the external simulator, it should simulate the same communicating entities as the external simulator scenario. The QualNet scenario should define the protocol stack and communication capabilities of all communicating entities in the scenario to be simulated.

You can create a QualNet scenario manually or use Extractor to generate the equivalent compatible QualNet scenario. Refer to *QualNet User's Guide* for details of creating QualNet scenarios. See Appendix A for details of using Extractor.

Tips for Configuring Scenarios

The following are some tips for creating scenarios for the HLA protocol:

- If terrain is used, specify a proper propagation model that will take terrain features into account, for example, the Irregular Terrain Model (ITM). See *Wireless Model Library* for details of ITM and other propagation models.

- The default parameter values for the 802.11 MAC protocol may prevent communication to take place over large distances. If the 802.11 MAC protocol is used in a scenario in which nodes are separated by large distances, make sure that the 802.11 MAC parameters are properly set. Alternatively, use another MAC protocol, such as Generic MAC. See *Wireless Model Library* for details of the 802.11 MAC and Generic MAC models.

2.2.2.1 Command Line Configuration

To enable the HLA protocol, include the following parameters in the scenario configuration (.config) file:

```
VRLINK            YES
VRLINK-PROTOCOL   <HLA-specification>
```

where

    `<HLA-specification>`   This is either `HLA13` (for the HLA 1.3 standard) or `HLA1516` (for the HLA 1516 standard).

Note: The default value of the parameter `VRLINK` is `NO`.

HLA Protocol Parameters

In addition to the other parameters required to configure a simulation scenario (refer to *QualNet User's Guide*), the parameters described in Table 2-1 must be configured in order to use the HLA protocol.

Note: If you want to run an HLA scenario created for QualNet 6.1 (or earlier), then you can modify that scenario to run with the current version of QualNet as described in Appendix C.

TABLE 2-1.   HLA Protocol Parameters

Parameter	Value	Description
`VRLINK-DEBUG` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Enables debug output from HLA communications effects requests.
`VRLINK-DEBUG-2` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Enables verbose debug output from HLA communications effects requests.
`VRLINK-FEDERATION-NAME` *Required* *Scope:* Global	String	Name of the HLA federation to join. All simulators in the joint simulation must use the same federation name.
`VRLINK-FEDERATE-NAME` *Required* *Scope:* Global	String	Name used by QualNet to identify itself to the HLA federation.

TABLE 2-1.   HLA Protocol Parameters (Continued)

Parameter	Value	Description
VRLINK-RPR-FOM-VERSION *Required* *Scope:* Global	List: ● 1.0 ● 2.0017 *Default:* 1.0	Version of RPR-FOM to use.
VRLINK-FED-FILE-PATH *Required* *Scope:* Global	Filename	If the HLA 1.3 standard is used, this is the name of the Federation Execution Details (.fed) file to use for the joint simulation. If the HLA 1516 standard is used, this is the name of the Federation Documentation Data (.xml) file to use for the joint simulation.
VRLINK-SITE-ID *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 1	Site ID used in the federation connection.
VRLINK-APPLICATION-ID *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 4000	Application ID at the site used in the federation connection. **Note:** The application ID should be unique among all federates.
VRLINK-XYZ-EPSILON *Optional* *Scope:* Global	Real *Range:* ≥ 0.0 *Default:* 0.5 *Unit:* meters	Minimum change in any of the X-, Y-, or Z-coordinates of an entity's position before the position is updated in QualNet.
VRLINK-TICK-INTERVAL *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 200MS	Minimum time between checks to see if the RTI has any Attribute Updates or Interactions for QualNet to process.
VRLINK-MOBILITY-INTERVAL *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 500MS	Minimum time between processing Entity position updates.
VRLINK-ENTITIES-FILE-PATH *Required* *Scope:* Global	Filename	Name of the Entities file. This file corresponds to the RPR FOM Physical Entity objects in HLA. The format of the Entities file is described in Section 2.2.2.1.1.
VRLINK-RADIOS-FILE-PATH *Required* *Scope:* Global	Filename	Name of the Radios file. This file corresponds to the RPR FOM Radio Transmitter objects in HLA. The format of the Radios file is described in Section 2.2.2.1.2.

TABLE 2-1.   HLA Protocol Parameters (Continued)

Parameter	Value	Description
`VRLINK-NETWORKS-FILE-PATH` *Required* *Scope:* Global	Filename	Name of the Networks file. This file defines groups of radios that can communicate with each other. The format of the Networks file is described in Section 2.2.2.1.2.
`VRLINK-HLA-DYNAMIC-STATISTICS` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `YES`	Indicates whether HLA dynamic statistics are enabled (see Section D.3 for details). HLA statistics will only be sent when using the GUI. Only the metrics selected using the Dynamics Statistics feature (refer to *QualNet User's Guide*) will be sent.
`VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE` *Optional* *Scope:* Global	List: • `VERBOSE` • `BRIEF` *Default:* `VERBOSE`	Level of detail in the Metric Update notifications (see Section D.3 for details). `VERBOSE`: Include nodeID and Metric Definition descriptions with the metric update. `BRIEF`: Include only the metric data. **Note:** This parameter is applicable only if `VRLINK-HLA-DYNAMIC-STATISTIS` is set to `YES`.
`VRLINK-HLA-DYNAMIC-STATISTICS-SEND-NODEID-DESCRIPTIONS` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether nodeID Description notifications are sent (see Section D.3 for details). **Note:** This parameter is applicable only if `VRLINK-HLA-DYNAMIC-STATISTIS` is set to `YES`.
`VRLINK-HLA-DYNAMIC-STATISTICS-SEND-METRIC-DEFINITIONS` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether Metric Definition notifications are sent (see Section D.3 for details). **Note:** This parameter is applicable only if `VRLINK-HLA-DYNAMIC-STATISTIS` is set to `YES`.

2.2.2.1.1 Format of the Entities File

The Entities file is used by both HLA and DIS (see Section 2.3). This file corresponds to the RPR FOM Physical Entity objects in HLA and to Entities in DIS.

Each line in the Entities file has the following format (all entries are on the same line):

```
<Marking>, <Force ID>, <Nationality>, <Lat>, <Long>, <Alt>, <Entity
Type Attributes>
```

where

`<Marking>`   Marking for the entity. This is a string value.

QualNet uses this string to identify the entity. Each entity in the federation should have a unique marking.

`<Force ID>`   Force ID of the entity. This can be `F` (for friendly), `O` (for opposing), or `N` (for neutral).

`<Nationality>`	Nationality of the entity. This is a string value.
	This is the same as the `Country` field of the `EntityType` attribute. See RPR FOM for HLA (see Section D.1) and the description of the Entity State PDU for DIS (see Section D.4).
`<Lat>`	Initial latitude of the entity.
`<Long>`	Initial longitude of the entity.
`<Alt>`	Initial altitude of the entity (in meters).
	This is the height above the WGS84 ellipsoid.
`<Entity Type Attributes>`	The seven fields `EntityType` attribute (which define the type of the unit):
	`<EA1>, <EA2>, <EA3>, <EA4>, <EA5>, <EA6>, <EA7>`
	where `<EA1>` through `<EA7>` are the seven fields of the `EntityType` attribute.

Example

The following is an example of an Entities file:

```
Tank11, F, USA, 43.533208,  6.680064,  0.5, 1, 1, 225, 1, 1, 1, 0
Tank12, F, USA, 43.533081,  6.676147,  0.5, 1, 1, 225, 1, 1, 1, 0
Tank13, F, USA, 43.533072,  6.671681,  0.5, 1, 1, 225, 1, 1, 1, 0
```

2.2.2.1.2 Format of the Radios File

The Radios file is used by both HLA and DIS (see Section 2.3). This file corresponds to the RPR FOM Radio Transmitter objects in HLA and to Transmitters in DIS.

Each line in the Radios file has the following format (all entries are on the same line):

```
<Node ID>, <Marking>, <Radio Index>, <Delta-X>, <Delta-Y>, <Delta-Z>,
<Radio Type Attributes>
```

where

`<Node ID>`	ID of the node corresponding to the Radio Transmitter object.
`<Marking>`	Marking for the host entity. This is a string value.
	This marking should be the same as the marking for the entity specified in the Entities file (see Section 2.2.2.1.1).
`<Radio Index>`	Radio index, which uniquely identifies the Radio Transmitter object on the host entity.
`<Delta-X>, <Delta-Y>, <Delta-Z>`	Relative position of the radio as mounted on the entity (in meters) using a world coordinate system that has the Y and Z axes swapped compared to the RPR-FOM geocentric Coordinate System.
`<Entity Type Attributes>`	The six fields `RadioType` attribute (which define the type of the radio):
	`<RA1>, <RA2>, <RA3>, <RA4>, <RA5>, <RA6>`
	where `<RA1>` through `<RA6>` are the six fields of the `RadioType` attribute.

Example

The following is an example of a Radios file:

```
1, Tank11, 0, 0.0, 0.0, 0.0, 7, 1, 225, 3, 1, 1
2, Tank12, 1, 0.0, 0.0, 0.0, 7, 1, 225, 3, 1, 1
3, Tank13, 2, 0.0, 0.0, 0.0, 7, 1, 225, 3, 1, 1
```

2.2.2.1.3 Format of the Networks File

The Networks file is used by both HLA and DIS (see Section 2.3). This file defines groups of radios that can communicate with each other.

Each line in the Networks file has the following format (all entries are on the same line):

```
<Network Name>; <Frequency>; <Node IDs>; <Destination Address>
```

where

`<Network Name>`	Name of the network.
	This field is not used by QualNet.
`<Frequency>`	Frequency of the network (in Hz).
	This field is not used by QualNet.
`<Node IDs>`	List of IDs of nodes that belong to the network. The node IDs are separated by commas.
	Note: A node can belong to only one network.
`<Destination Address>`	IP address of the default destination used for transmissions on the network. The default destination is used when the receiver is not specified in the communications request.
	The local broadcast IP address (`255.255.255.255`) is the default, although other IP addresses can be substituted (e.g., multicast IP addresses). A value of `0.0.0.0` indicates that all transmissions on the network should be modeled as unicast (in which case the default destination is dynamically assigned at run time).
	Note: The default address, `255.255.255.255`, is used for single-hop broadcasts. `0.0.0.0` should generally be used for multi-hop (or otherwise relayed) communications, when a network spans network infrastructure present in QualNet but not represented in HLA.

Example

The following is an example of a Networks file:

```
N1;30000000;1,2,3,4;255.255.255.255
N2;35000000;5,6,7,8;255.255.255.255
N3;40000000;9,10,11,12;255.255.255.255
N4;45000000;13,14,15,16;255.255.255.255
N5;50000000;17,18,19,20;255.255.255.255
N6;225000000;21,23;255.255.255.255
N7;230000000;24,27;0.0.0.0
```

2.2.2.2 GUI Configuration

In addition to the other parameters required to configure a simulation scenario (refer to *QualNet User's Guide*), the HLA protocol parameters must be configured as described below.

To configure the HLA protocol parameters, perform the following steps:

1. Go to **Scenario Properties Editor > External Interfaces > VR-Link Interface**.

2. Set **VR-Link Interface** to *Yes*.

3. Set **Protocol** to *HLA13* or *HLA1516* and set the dependent parameters listed in Table 2-2.

FIGURE 2-1.   Configuring HLA Protocol Parameters

TABLE 2-2.   Command Line Equivalent of HLA Protocol Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Enable Debugging	Global	VRLINK-DEBUG
Enable Verbose Debugging	Global	VRLINK-DEBUG-2
Federation Name	Global	VRLINK-FEDERATION-NAME
Federate Name	Global	VRLINK-FEDERATE-NAME
RPR FOM Version	Global	VRLINK-RPR-FOM-VERSION
Federation File	Global	VRLINK-FED-FILE-PATH
Site ID	Global	VRLINK-SITE-ID
Application Number	Global	VRLINK-APPLICATION-NUMBER
XYZ Epsilon	Global	VRLINK-XYZ-EPSILON
Tick Interval	Global	VRLINK-TICK-INTERVAL
Mobility Interval	Global	VRLINK-MOBILITY-INTERVAL
Entities File	Global	VRLINK-ENTITIES-FILE-PATH
Radios File	Global	VRLINK-RADIOS-FILE-PATH
Network File	Global	VRLINK-NETWORKS-FILE-PATH
Enable VR-Link Statistics	Global	VRLINK-HLA-DYNAMIC-STATISTICS

Setting Parameters

- Set **Federation File** to the name of the Federation Execution Details (.fed) file (for HLA 1.3) or the Federation Documentation Data (.xml) file (for HLA 1516).

- Set **Entities File** to the name of the Entities file. See Section 2.2.2.1.1 for the format of the Entities file.

- Set **Radios File** to the name of the Radios file. See Section 2.2.2.1.2 for the format of the Radios file.

- Set **Networks File** to the name of the Networks file. See Section 2.2.2.1.3 for the format of the Networks file.

4. If **Enable VR-Link Statistics** is set to *Yes*, then set the dependent parameters listed in Table 2-2.
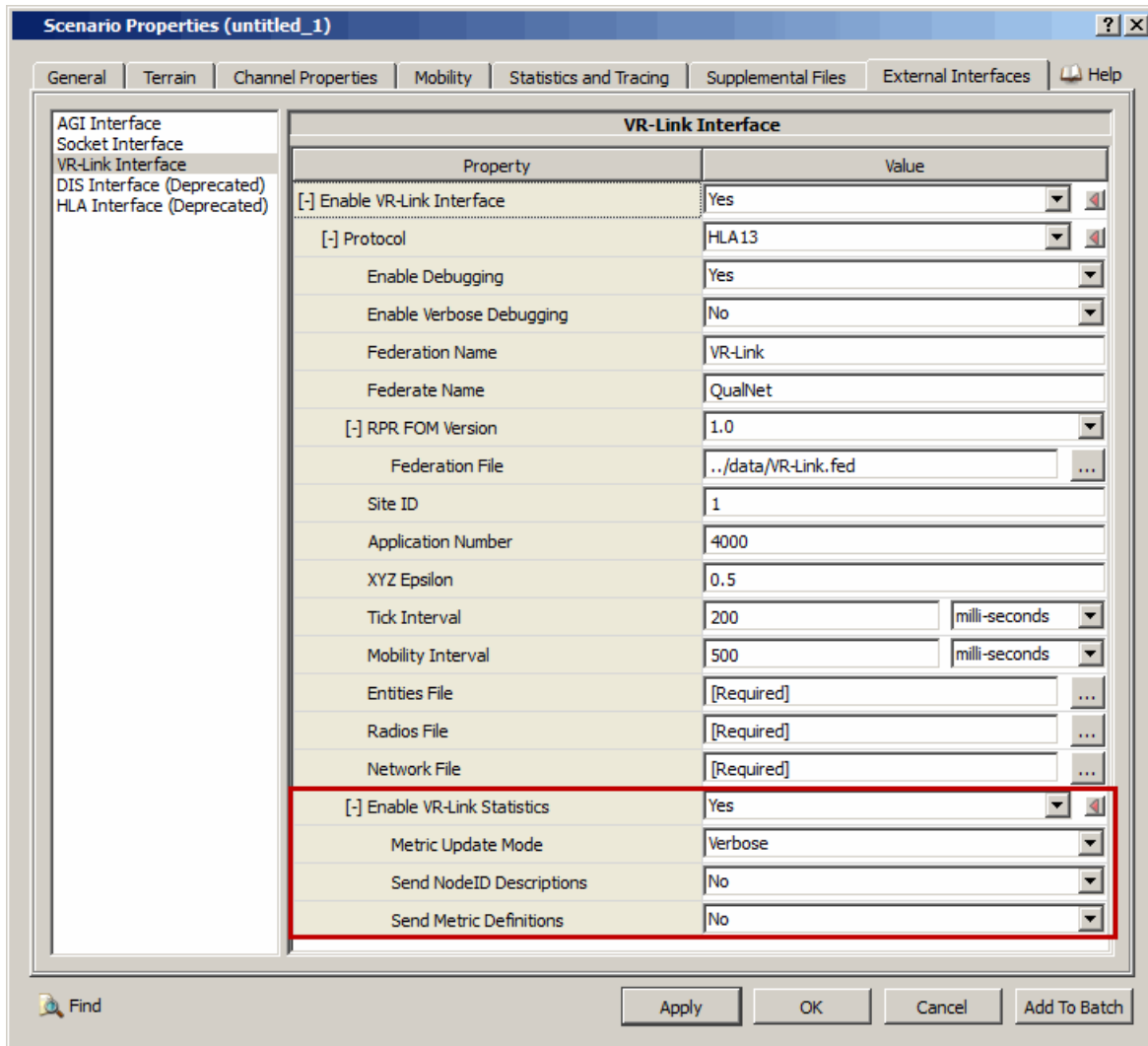
FIGURE 2-2.   Configuring HLA Statistics Parameters

TABLE 2-3.   Command Line Equivalent of HLA Statistics Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Metric Update Mode	Global	`VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE`
Send NodeID Descriptions	Global	`VRLINK-HLA-DYNAMIC-STATISTICS-SEND-NODEID-DESCRIPTIONS`
Send Metric Definitions	Global	`VRLINK-HLA-DYNAMIC-STATISTICS-SEND-METRIC-DEFINITIONS`

2.2.3 Sequence of Events

The sequence of events in a typical HLA federation consisting of QualNet and a client federate is as follows:

1. Client loads the scenario and joins federation.

2. QualNet loads the equivalent scenario and joins federation.

3. QualNet waits for the first entity update sent by the client federate. The update can be for any entity, whether or not it has a representation in the QualNet scenario. (Only communicating entities in the client scenario need to be represented in QualNet.)

4. The QualNet simulation starts initializing when the first entity update is received. The initialization time depends on the complexity of the scenario. The simulation starts after the initialization is complete. The warm-up phase of QualNet starts when the simulation starts. In the warm-up phase, depending on the configuration, all communication effects requests are treated as being either unsuccessful or successful with zero delay.

5. During the simulation, when the client federate moves an entity, it updates the Entities position and the RTI sends an Entity Attribute Update (see Section D.1). When QualNet receives the Entity Attribute Update, it consults the Entities and Radios files (see Section 2.2.2) to move the appropriate QualNet nodes.

6. During the simulation, the client federate sends ApplicationSpecificRadioSignal (ASRS) interactions to request communication effects modeling (see Section D.1). When QualNet receives an ASRS interaction, it models the traffic and can respond as follows:

   • If the communication is successful, QualNet sends a Process Message interaction indicating that the message was delivered to the destination. This interaction is sent to the client as soon as the modeled message is delivered to the destination node in QualNet.

   • Whether or not the communication is successful, QualNet sends a Timeout interaction summarizing the delivery status to all potential recipients. This occurs when the processing of the message is complete: either after the timeout value indicated in the original ApplicationSpecificRadioSignal interaction, or after all potential recipients have processed the signal, whichever occurs first.

7. The client federate can also indicate if entities are damaged or destroyed, or hosted radios are turned on or off. QualNet will model such effects appropriately (for example, a destroyed entity will deactivate all associated QualNet nodes).

8. The QualNet simulation runs for the time specified by the QualNet SIMUALTION-TIME parameter, irrespective of when the client simulation ends. The statistics file is generated when the QualNet simulation ends.

2.2.4 Using HLA Protocol with testfed

This section describes how to use the QualNet HLA protocol with testfed. testfed (test federate) is a command-line based program included in the QualNet distribution which is primarily used to test the QualNet HLA protocol without the overhead of running a full constructive simulation tool. See Appendix B for details.

2.2.4.1 Creating Scenarios for testfed and QualNet

testfed uses the same scenario files as QualNet. Refer to *QualNet User's Guide* for details of configuring scenarios in QualNet. In addition, the parameters described in Section 2.2.2 also need to be configured.

2.2.4.2 testfed-QualNet Demonstration

This section describes how to run testfed with QualNet using a sample scenario, unicast, located in QUALNET_HOME/scenarios/vrf/unicast.

To run this demo, an RTI must already be installed, and the environment variables must be set as described in Section 2.1.3.

To run the demo, do the following:

1. If you are using MAK RTI make sure that rtiAssistant is already running.

2. Start RtiExec, by doing the following (the following steps assume that MAK RTI is installed at the default location):

    a. Select **Start > All Programs > MAK Technologies > MAK RTI 4.1.1 > rtiexec**.
    b. Select rtiexec instance that you wish to use.
    c. Wait for rtiexec to finish initializing.

3. Start the QualNet GUI and open the QualNet scenario.
    - If you are using HLA 1.3, then open QUALNET_HOME/scenarios/vrf/unicast/unicast-hla.config.
    - If you are using HLA 1516, then open QUALNET_HOME/scenarios/vrf/unicast/unicast-hla1516.config.

4. Open the **Output Window** panel to display the status and HLA debug messages. Click on the **Run Simulation** button to begin simulation initialization.

5. Open a command window and change the directory to QUALNET_HOME/scenarios/vrf/unicast.

    To run testfed with HLA 1.3, type the following command:

    ```
    testfed -P HLA13 -F ../data/VR-Link.fed unicast
    ```

    To run testfed with HLA 1516, type the following command:

    ```
    testfed -P HLA1516 -F ../data/VR-Link.xml unicast
    ```

6. Press the **Play** button in the QualNet GUI. Let the scenario run till completion.

Scenario Behavior

The testfed and QualNet scenarios run simultaneously and interact with each other. The file unicast.testfed (see Figure 2-3) in the directory QUALNET_HOME/scenarios/vrf/unicast contains commands that testfed sends to QualNet. (See Section B.5 for details of testfed commands.)

```
Register 5S
Move 10S 5 0.01 0.01 20
Damage 12S 4 2
TxState 13S 4 2
Orientation 14S 5 1.1 1.1 1
#Basic Unicast
SendCommRequest 15S 1 24 10 2
#Voice Unicast
SendCommRequest 20S 1 V3 30 3
Velocity 30S 5 0 100 100
Velocity 50S 5 0 0 0
Quit 60S
```

FIGURE 2-3. File unicast.testfed

When testfed is started, the testfed window displays all the commands contained in the file unicast.testfed.

testfed sends commands contained in the file unicast.tesfed to the QualNet simulation at the times associated with the commands. Each command is displayed in the testfed window when it is sent to the QualNet simulation. Any responses received from the QualNet simulation are also displayed in the testfed window.

The QualNet simulation processes the commands received from testfed. Debug messages are displayed in the **Output Window**. QualNet also sends messages back to testfed in response to some of the commands received from testfed.

Some of the commands and their effects are described below.

- `Register 5S`

  At 5 seconds, testfed sends a command to register the entities and radios. In response, QualNet maps the entities and radios to QualNet nodes.

- `Move 10S 5 0.01 0.01 20`

  At 10 seconds, testfed sends a command to move entity 5. In QualNet, node 5 is moved to the specified location.

- `SENDCOMMREQUEST 15S 1 24 10 2`

  At 15 seconds, testfed sends a communication effects request to QualNet. In QualNet, node 1 sends a 24 byte message to node 2. Since the message is delivered successfully, QualNet sends a Process Message Notification followed by a Timeout Notification. The contents of the notifications sent by QualNet to testfed are displayed in the testfed window.

- `Velocity 30S 5 0 100 100`

  At 30 seconds, testfed sends a command to QualNet to change the velocity of entity 5. In QualNet, node 5 starts moving with the velocity (speed and direction) specified in the command. In the **Output Window**, QualNet periodically displays the coordinates of node 5 as it moves. This continues until the next command (`Velocity 50S 5 0 0 0`) is received, which causes node 5 to stop moving at 50 seconds.

- `Quit 60S`

  At 60 seconds, testfed sends a quit command to QualNet. The testfed simulation ends, but the QualNet simulation continues until the scenario simulation time (180 seconds).

2.2.5 Using HLA Protocol with VR Forces

This section describes how to use the QualNet HLA protocol with MAK's VR-Forces (http://www.mak.com).

VR-Forces is a Computer-Generated Forces (CGF) toolkit. It provides an intuitive GUI that allows non-programmers to build scenarios by positioning forces, creating routes and waypoints, and assigning tasks or plans with simple point and click. During scenario execution, VR-Forces vehicles interact with the terrain, follow roads, avoid obstacles, detect and engage enemy forces, and calculate damage. Vehicle dynamics, sensor capabilities, and damage models are fully configurable with no programming necessary. Entities in VR-Forces also communicate with each other, such as sending and receiving commands, providing sensor updates, and other network centric operations. When run by itself, a VR-Forces simulation assumes perfect communications, i.e., all messages are delivered, regardless of the communications environment.

In this section, we describe how to run QualNet with VR-Forces using HLA by means of a demo.

2.2.5.1 Creating Scenarios

Create scenarios for VR-Forces and QualNet, as described in Section 2.2.2.

In this demo, we will run the CommsDemo scenario in VR-Forces and QualNet. Files for this scenario are located in QUALNET_HOME/scenarios/vrf/CommsDemo.

2.2.5.2 Starting RTI

In order to run the HLA protocol, the RTI software should be installed and properly configured.

This section describes how to start the MAK RTI. For details, refer to *MAK RTI Reference Manual* and *VR-Forces User's Guide.* (For other RTIs, refer to the documentation for the RTI.)

For this demo, start the RTI as follows:

1. If you are using MAK RTI make sure that rtiAssistant is already running.

2. Start RtiExec, by doing the following (the following steps assume that MAK RTI is installed at the default location):

    a. Select **Start > All Programs > MAK Technologies > MAK RTI 4.1.1 > rtiexec**.

    b. Select rtiexec instance that you wish to use.

    c. Wait for rtiexec to finish initializing.

2.2.5.3 Starting VR-Forces

This section outlines the procedure to start VR-Forces and load a scenario. For more details, refer to *VR-Forces User's Guide*.

For this demo, start VR-Forces as follows:

1. Select **Start > All Programs > MAK Technologies > VR-Forces 4.0.4 > VR-Forces GUI + Simulation Engine**.

2. In the **Simulations Connections Configuration** dialog that opens up (see Figure 2-4), do the following:

- In the left panel, select *HLA 1.3 RPR 1.0* as the connection to use.

- Set **Federation Name** and **RPR FOM Version** to the same values as the VRLINK-FEDERATION-NAME and VRLINK-RPR-FOM-VERSION parameters, respectively, in the QualNet scenario configuration file (these correspond to **Federation Name** and **RPR FOM Version**, respectively, in the QualNet GUI).
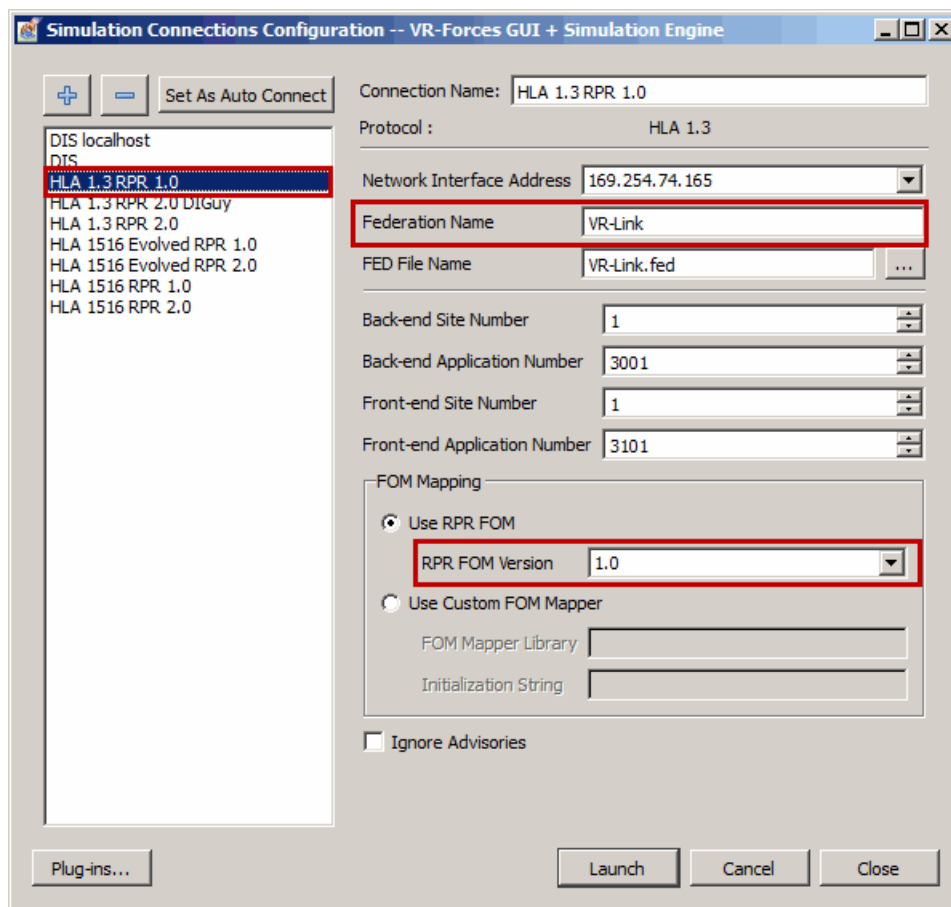
FIGURE 2-4.   VR Forces Simulation Connections Configuration for HLA

3. Click on the **Launch** button to start VR Forces.

4. If you have not configured auto connections for the RTI, then in the RTI connection window that is displayed, select an RTI connection and click **Connect**.

> **Note:**   All simulators in the federation must use the same RTI instance (rtiExec). Select an RTI connection that will also be available to the computer running QualNet.

5. Enable external communications. To do this, go to the **Settings** menu and select **Use External Communication Model**. (This needs to be done only when you are using VR Forces for the first time.)

6. Load the scenario to be simulated. Refer to *VR-Forces User's Guide* for details. For this demo, load the file QUALNET_HOME/scenarios/vrf/CommsDemo/CommsDemo.scn.

> **Note:**   Do not press the **Play** button. The simulation should be started after the QualNet scenario has also been loaded, as described in .

2.2.5.4 Starting QualNet Simulation

This section outlines the procedure of starting a simulation in QualNet. For details of using QualNet, refer to *QualNet User's Guide*.

> **Note:**   This section assumes that RtiExec has already been started, as described in Section 2.2.5.2.

Starting Simulation from the Command Line

To start a QualNet simulation from the command line, perform the following steps.

1. Open a command window and change the directory to the folder where the QualNet scenario (which is equivalent to the VR-Forces scenario) is located. For this demo, the scenario is located in the folder QUALNET_HOME/scenarios/vrf/CommsDemo.

2. Enter the following command to run the scenario:

   ```
   qualnet <config-file>
   ```

   where `<config-file>` is the name of the scenario configuration (.config) file.

   If you are using HLA 1.3, the configuration file is CommsDemo-hla.config. If you are using HLA 1516, the configuration file is CommsDemo-hla1516.config.

   Status messages are displayed in the command window.

3. If you have not configured auto connections for the RTI, then in the RTI connection window that is displayed, select an RTI connection and click **Connect**.

Starting Simulation from the GUI

To start a QualNet simulation from the GUI, perform the following steps.

1. Start the QualNet GUI.

2. In the **File System** panel, navigate to the QualNet configuration (.config) file for the scenario (which is equivalent to the VR-Forces scenario), right-click and select **Open**. For this demo, the configuration files are located in the folder QUALNET_HOME/scenarios/vrf/CommsDemo. If you are using HLA 1.3, the configuration file is CommsDemo-hla.config. If you are using HLA 1516, the configuration file is CommsDemo-hla1516.config.

3. Click on the **Run Simulation** button to begin simulation initialization. Status messages are displayed in the **Output Window** panel.

4. After the scenario has finished initializing, click the **Play** button to start the simulation.

5. If you have not configured auto connections for the RTI, then in the RTI connection window that is displayed, select an RTI connection and click **Connect**.

2.2.5.5 Running the Joint Simulation Exercise

To simulate the scenario jointly in VR-Forces and QualNet, press the **Play** button in VR-Forces. Visualization of the scenario should progress in both VR-Forces and QualNet.

Expected Behavior

In this scenario, one of the tanks tells the UAV to turn on its sensors and look for the enemy. The UAV then moves towards the enemy and reports their location to the tanks. The two tanks then move towards the enemy and destroy them.

Re-running the Simulation

To run the simulation again, do the following:

1. If the QualNet simulation is still running, stop it by pressing the **Stop** button in the QualNet GUI or pressing **Ctrl+C** in the command line window.

2. In VR-Forces, press the **Pause** button and then the **Rewind** button.

3. Restart the simulation in QualNet, as described in Section 2.2.5.4.

4. In VR-Forces, press the **Play** button.

2.3 Distributed Interaction Simulation (DIS) Protocol

Distributed Interaction Simulation (DIS) is an IEEE standard for interfacing multiple simulation tools into a single, real-time exercise. The transport of information between simulators is performed using UDP and broadcast and/or multicast IP. Although formally superseded by HLA and IEEE 1516, DIS still remains popular today for its simplicity of operation and the ease in which a DIS interface can be created.

The QualNet DIS protocol supports DIS protocol versions 4, 5, and 6.

> **Note:** The QualNet DIS model does not support dynamic statistics.

Using the DIS Protocol

Section 2.3.1 describes the general procedure for using the DIS protocol.

Section 2.2.2 describes how to create QualNet scenarios for use with the DIS protocol.

Section 2.3.3 describes the sequence of events in an exercise consisting of QualNet acting as a communications server and another simulation application acting as a communications client.

Section 2.3.5 describes how to use the DIS protocol to run a simulation in VR Forces and QualNet.

Appendix D describes the DIS Protocol Data Units and the DIS Extensions Interface Communications Document.

2.3.1 General Procedure for Using DIS Protocol

In general, simulating a scenario using the DIS protocol comprises the steps listed below. These steps are explained in the following sections.

1. Create a simulation scenario for the external simulator.
2. Create a QualNet scenario that is compatible with the external simulation scenario.
3. Start the external simulator and load the scenario.
4. Start QualNet and run the QualNet scenario equivalent to the external simulator scenario.
5. Run the simulation in the external simulator.

2.3.2 Creating Scenarios

For QualNet to provide communication effects to the external simulator, it should simulate the same communicating entities as the external simulator scenario. The QualNet scenario should define the protocol stack and communication capabilities of all communicating entities in the scenario to be simulated.

2.3.2.1 Command Line Configuration

To enable the DIS protocol, include the following parameter in the scenario configuration (.config) file:

```
VRLINK            YES
VRLINK-PROTOCOL   DIS
```

> **Note:** The default value of the parameter `VRLINK` is `NO`.

DIS Protocol Parameters

In addition to the other parameters required to configure a simulation scenario (refer to *QualNet User's Guide*), the parameters described in Table 2-1 must be configured in order to use the DIS protocol.

> **Note:** If you want to run an DIS scenario created for QualNet 6.1 (or earlier), then you can modify that scenario to run with the current version of QualNet as described in Appendix C.

TABLE 2-4.   DIS Protocol Parameters

Parameter	Value	Description
VRLINK-DEBUG-PRINT-COMMS *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables debug output from DIS communications effects requests.
VRLINK-DEBUG-PRINT-COMMS-2 *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables verbose debug output from DIS communications effects requests.
VRLINK-DEBUG-PRINT-MAPPING *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables printing of the mapping of Entity State PDUs to QualNet nodes, as it occurs.
VRLINK-DEBUG-PRINT-DAMAGE *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables printing of information to the screen when the Damage field for an entity changes. The Damage field appears in the Entity Appearance record of the Entity State PDU.
VRLINK-DEBUG-PRINT-TX-STATE *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables printing of transmitter state (on/off) to the screen.
VRLINK-DEBUG-PRINT-TX-POWER *Optional* *Scope:* Global	List: • YES • NO *Default:* YES	Enables printing of details when QualNet probabilistically decreases the maximum transmit power of hosted radios when an entity suffers damage. **Note:** An entity with a Damage field equal to Destroyed will have all radios turned off.

TABLE 2-4.   DIS Protocol Parameters (Continued)

Parameter	Value	Description
VRLINK-DEBUG-PRINT-MOBILITY *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Enables printing of information to the screen when an entity is moved within QualNet.
VRLINK-DEBUG-PRINT-TRANSMITTER-PDU *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Enables printing of information about received Transmitter PDUs to the screen.
VRLINK-DEBUG-PRINT-PDUS *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Enables printing of a hex dump of most DIS PDUs.
VRLINK-DIS-IP-ADDRESS *Optional* *Scope:* Global	IP Address *Default:* 127.255.255.255	IP address on which QualNet receives and sends DIS PDUs. This can be set to a local broadcast address (127.255.255.255) or a multicast address.
VRLINK-DIS-NETWORK-INTERFACE *Optional* *Scope:* Global	IP Address *Default:* 127.0.0.1	IP address of the network interface for the socket to use.
VRLINK-DIS-SUBNET-MASK *Optional* *Scope:* Global	IP Address	Subnet mask for the DIS socket used for listening and sending.
VRLINK-DIS-PORT *Optional* *Scope:* Global	Integer *Range:* [0, 65535] *Default:* 3000	UDP port on which QualNet receives and sends DIS PDUs.
VRLINK-EXERCISE-ID *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 1	Exercise ID used by the other federates in the exercise.
VRLINK-SITE-ID *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 1	Site ID used in the federation connection.

TABLE 2-4.   DIS Protocol Parameters (Continued)

Parameter	Value	Description
VRLINK-APPLICATION-ID *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 4000	Application ID at the site used in the federation connection. **Note:** The application ID should be unique among all federates.
VRLINK-MULTICAST-TTL *Optional* *Scope:* Global	Integer *Range:* [0, 255] *Default:* 2	Multicast time-to-live for the DIS socket.
VRLINK-RECEIVE-DELAY *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 200MS	Minimum amount of time that needs to pass since the last successful call to recv() to check for DIS PDUs on the socket, before recv() is called again.
VRLINK-MAX-RECEIVE-DURATION *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 5MS	Maximum amount of time QualNet waits to empty the socket. Normally, QualNet calls recv() until there are no pending DIS PDUs on the socket (the DIS protocol uses non-blocking sockets). Theoretically for very busy networks the socket will never empty, or it may take a long time to empty the socket. This parameter limits this duration, so as to give control back to the QualNet simulation kernel and also to allow time for other external interfaces.
VRLINK-XYZ-EPSILON *Optional* *Scope:* Global	Real *Range:* ≥ 0 *Default:* 0.5 *Unit:* meters	Minimum change in any of the x, y, or z coordinates of the GCC position before the change is reflected in QualNet.
VRLINK-MOBILITY-INTERVAL *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 500MS	Minimum amount of time that needs to pass since the last scheduled mobility event for a given entity. Setting this value to 0S will cause almost no mobility updates to be skipped. Setting it to a large value generally reduces simulation time, but skips more updates.
VRLINK-ENTITIES-FILE-PATH *Required* *Scope:* Global	Filename	Name of the Entities file. This file corresponds to Entities in DIS. The format of the Entities file is described in Section 2.2.2.1.1.
VRLINK-RADIOS-FILE-PATH *Required* *Scope:* Global	Filename	Name of the Radios file. This file corresponds to Transmitters in DIS. The format of the Radios file is described in Section 2.2.2.1.2.

TABLE 2-4.   DIS Protocol Parameters (Continued)

Parameter	Value	Description
`VRLINK-NETWORKS-FILE-PATH` *Required* *Scope:* Global	Filename	Name of the Networks file. This file defines groups of radios that can communicate with each other. The format of the Networks file is described in Section 2.2.2.1.3.
`VRLINK-DISABLE-REFLECTED-RADIO-TRANSMITTER-TIMEOUTS` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Disables timeouts of reflected radio transmitter objects in QualNet. If this parameter is set to `NO`, radios are removed 12 seconds after receiving the last received transmitter PDU for a given radio. **Note:** This parameter should be set to `YES` if a publishing federate does not send Transmitter PDU heartbeats.

2.3.2.2 GUI Configuration

In addition to the other parameters required to configure a simulation scenario (refer to *QualNet User's Guide*), the DIS protocol parameters must be configured as described below.

To configure the DIS protocol parameters, perform the following steps:

1. Go to **Scenario Properties Editor > External Interfaces > VR-Link Interface**.

2. Set **VR-Link Interface** to *Yes*.

3. Set **Protocol** to *DIS* and set the dependent parameters listed in Table 2-5.

FIGURE 2-5.   Configuring DIS Protocol Parameters

TABLE 2-5.   Command Line Equivalent of DIS Protocol Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Debug: Print Comms	Global	`VRLINK-DEBUG-PRINT-COMMS`
Debug: Print Comms Verbose	Global	`VRLINK-DEBUG-PRINT-COMMS-2`
Debug: Print Mapping	Global	`VRLINK-DEBUG-PRINT-MAPPING`
Debug: Print Damage	Global	`VRLINK-DEBUG-PRINT-DAMAGE`
Debug: Print TX State	Global	`VRLINK-DEBUG-PRINT-TX-STATE`
Debug: Print TX Power	Global	`VRLINK-DEBUG-PRINT-TX-POWER`
Debug: Print Mobility	Global	`VRLINK-DEBUG-PRINT-MOBILITY`
Debug: Print Transmitter PDU	Global	`VRLINK-DEBUG-PRINT-TRANSMITTER-PDU`
Debug: Print PDUs	Global	`VRLINK-DEBUG-PRINT-PDUS`
Destination IP Address	Global	`VRLINK-DIS-IP-ADDRESS`
Network Interface Address	Global	`VRLINK-DIS-NETWORK-INTERFACE`
Specify Subnet Mask	Global	N/A
Port	Global	`VRLINK-DIS-PORT`
Exercise ID	Global	`VRLINK-DIS-EXERCISE-ID`
Site ID	Global	`VRLINK-SITE-ID`
Application Number	Global	`VRLINK-APPLICATION-NUMBER`
Multicast TTL	Global	`VRLINK-MULTICAST-TTL`
Receive Delay	Global	`VRLINK-RECEIVE-DELAY`
Max Receive Duration	Global	`VRLINK-MAX-RECEIVE-DURATION`
XYZ Epsilon	Global	`VRLINK-XYZ-EPSILON`
Mobility Interval	Global	`VRLINK-MOBILITY-INTERVAL`
Entities File	Global	`VRLINK-ENTITIES-FILE-PATH`
Radios File	Global	`VRLINK-RADIOS-FILE-PATH`
Networks File	Global	`VRLINK-NETWORKS-FILE-PATH`
Disable Reflected Radio Transmitter Timeouts	Global	`VRLINK-DISABLE-REFLECTED-RADIO-TRANSMITTER-TIMEOUTS`

Setting Parameters

- To specify a subnet mask for the DIS socket, set **Specify Subnet Mask** to *Yes*.

- Set **Entities File Path** to the name of the Entities file. See Section 2.2.2.1.1 for the format of the Entities file.

- Set **Radios File Path** to the name of the Radios file. See Section 2.2.2.1.2 for the format of the Radios file.

- Set **Networks File Path** to the name of the Networks file. See Section 2.2.2.1.3 for the format of the Networks file.

4. If **Specify Subnet Mask** is set to *Yes*, then set the dependent parameters listed in Table 2-6.
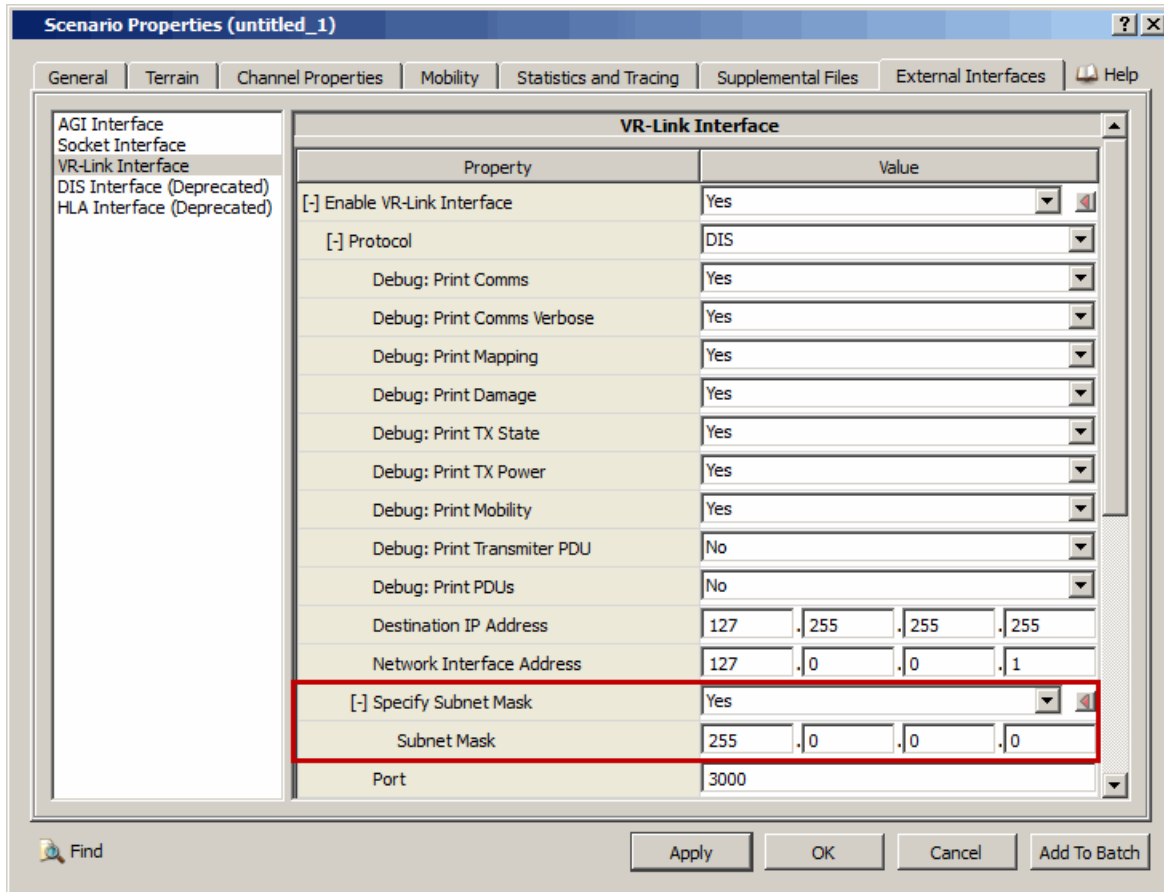
FIGURE 2-6. **Specifying Subnet Mask**

TABLE 2-6. **Command Line Equivalent of Subnet Mask Parameters**

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Subnet Mask	Global	VRLINK-DIS-SUBNET-MASK

2.3.3 Sequence of Events

The sequence of events in a typical DIS exercise consisting of QualNet acting as a communications server and another simulation application acting as a communications client is as follows:

1. The client loads the scenario and joins the exercise.

2. QualNet loads the equivalent scenario and joins the exercise.

3. QualNet waits for the first entity update sent by the client. The update can be for any entity, whether or not it has a representation in the QualNet scenario. (Only communicating entities in the external simulator scenario need to be represented in QualNet.)

4. The QualNet simulation starts initializing when the first entity update is received. The initialization time depends on the complexity of the scenario. The simulation starts after the initialization is complete. The warm-up phase of QualNet starts when the simulation starts. In the warm-up phase, depending on the

configuration, all communication effects requests are treated as being either unsuccessful or successful with zero delay.

5. During the simulation, when the client simulation moves an entity, it sends an Entity State PDU (see Section D.4). When QualNet receives the Entity State PDU (see Section D.4), it consults the Entities and Radios files (see Section 2.2.2) to move the appropriate QualNet nodes.

6. During the simulation, the client sends Signal PDUs to request communication effects modeling (see Section D.4). When QualNet receives a Signal PDU, it models the traffic and can respond as follows:

   • If the communication is successful, QualNet sends a Data PDU containing a Process Message result indicating that the message was delivered to the destination. This PDU is sent to the client as soon as the modeled message is delivered to the destination node in QualNet.

   • Whether or not the communication is successful, QualNet sends a Data PDU containing a Timeout result summarizing the delivery status to all potential recipients. This occurs when the processing of the message is complete: either after the timeout value indicated in the original Signal PDU, or after all potential recipients have processed the signal, whichever occurs first.

7. The client can also indicate if entities are damaged or destroyed, or hosted radios are turned on or off. QualNet will model such effects appropriately (for example, a destroyed entity will deactivate all associated QualNet nodes).

8. The QualNet simulation runs for the time specified by the QualNet SIMUALTION-TIME parameter, irrespective of when the external simulation ends. The statistics file is generated when the QualNet simulation ends.

2.3.4 Using DIS Protocol with testfed

This section describes how to use the QualNet DIS protocol with testfed. testfed (test federate) is a command-line based program included in the QualNet distribution which is primarily used to test the QualNet HLA protocol without the overhead of running a full constructive simulation tool. See Appendix B for details.

2.3.4.1 Creating Scenarios for testfed and QualNet

testfed uses the same scenario files as QualNet. Refer to *QualNet User's Guide* for details of configuring scenarios in QualNet. In addition, the parameters described in Section 2.2.2 also need to be configured.

2.3.4.2 testfed-QualNet Demonstration

This section describes how to run testfed with QualNet using a sample scenario, unicast, located in QUALNET_HOME/scenarios/vrf/unicast.

To run the demo, do the following:

1. Start the QualNet GUI and open the scenario QUALNET_HOME/scenarios/vrf/unicast-dis in the GUI. Open the **Output Window** to display the status and DIS debug messages. Click on the **Run Simulation** button to begin simulation initialization.

2. Open a command window and change the directory to QUALNET_HOME/scenarios/vrf/unicast. Type the following command:

   ```
   testfed -P DIS unicast
   ```

3. Press the **Play** button in the QualNet GUI. Let the scenario run till completion.

Scenario Behavior

The behavior of the unicast scenario when testfed is run with QualNet using DIS is the same as the scenario behavior when testfed is run with QualNet using HLA. See Figure 2.2.4.2 for details.

2.3.5 VR-Forces/QualNet Demonstration

This section describes how to use the QualNet DIS protocol with MAK's VR-Forces (http://www.mak.com).

VR-Forces is a Computer-Generated Forces (CGF) toolkit. It provides an intuitive GUI that allows non-programmers to build scenarios by positioning forces, creating routes and waypoints, and assigning tasks or plans with simple point and click. During scenario execution, VR-Forces vehicles interact with the terrain, follow roads, avoid obstacles, detect and engage enemy forces, and calculate damage. Vehicle dynamics, sensor capabilities, and damage models are fully configurable with no programming necessary. Entities in VR-Forces also communicate with each other, such as sending and receiving commands, providing sensor updates, and other network centric operations. When run by itself, a VR-Forces simulation assumes perfect communications, i.e., all messages are delivered, regardless of the communications environment.

In this section, we describe how to run QualNet with VR-Forces using DIS by means of a demo.

2.3.5.1 Creating Scenarios

Create scenarios for VR-Forces and QualNet, as described in Section 2.2.2.

In this demo, we will run the CommsDemo scenario in VR-Forces and QualNet. Files for this scenario are located in QUALNET_HOME/scenarios/vrf/CommsDemo.

2.3.5.2 Starting VR-Forces

Perform the following steps to start VR-Forces. For more details, refer to *VR-Forces User's Guide*.

1. Select **Start > All Programs > MAK Technologies > VR-Forces 4.0.4 > VR-Forces GUI + Simulation Engine**.

2. In the **Simulations Connections Configuration** dialog that opens up (see Figure 2-7), do the following:

    *   Select a connection in the left panel:
        -   If VR-Forces and QualNet are run on the same machine, then select *DIS localhost*.
        -   If VR-Forces and QualNet are run on different machines, then select *DIS*.

    *   Set **Port** and **Exercise ID** to the same values as the `VRLINK-DIS-PORT` and `VRLINK-EXERCISE-ID` parameters, respectively, in the QualNet scenario configuration file (these correspond to **DIS Port ID** and **DIS Exercise ID**, respectively, in the QualNet GUI).

    *   Set **Destination Address** to the same value as the `VRLINK-DIS-IP-ADDRESS` parameter in the QualNet scenario configuration file (this corresponds to **DIS IP Address** in the QualNet GUI).

    *   If VR-Forces and QualNet are run on the same machine, then set **Network Interface Address** to the same value as the `VRLINK-DIS-NETWORK-INTERFACE` parameter in the QualNet scenario configuration file (this corresponds to **DIS Network Interface Address** in the QualNet GUI).
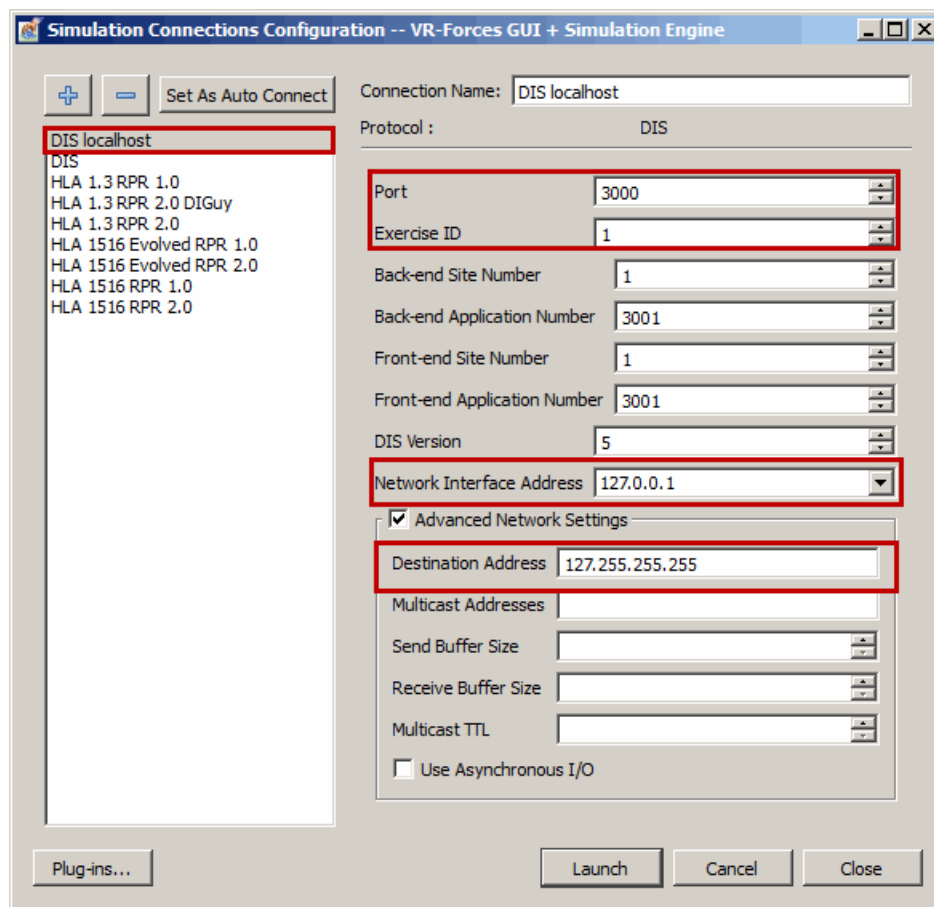
FIGURE 2-7.   VR Forces Simulation Connections Configuration for DIS

3. Click on the **Launch** button to start VR Forces.

4. Enable external communications. To do this, go to the **Settings** menu and select **Use External Communication Model**. (This needs to be done only when you are using VR Forces for the first time.)

5. Load the scenario to be simulated. Refer to *VR-Forces User's Guide* for details. For this demo, load the file QUALNET_HOME/scenarios/vrf/CommsDemo/CommsDemo.scn.

   Note: Do not press the **Play** button. The simulation should be started after the QualNet scenario has also been loaded, as described in .

2.3.5.3 Starting QualNet Simulation

This section outlines the procedure of starting a simulation in QualNet. For details of using QualNet, refer to *QualNet User's Guide*.

Starting Simulation from the Command Line

To start a QualNet simulation from the command line, perform the following steps.

1. Open a command window and change the directory to the folder where the QualNet scenario (which is equivalent to the VR-Forces scenario) is located. For this demo, the scenario is located in QUALNET_HOME/scenarios/vrf/CommsDemo.

2. Enter the following command to run the scenario:

   ```
   qualnet <config-file>
   ```

   where `<config-file>` is the name of the scenario configuration (.config) file.

   For this demo, the configuration file is CommsDemo-dis.config.

   Status messages are displayed in the command window.

Starting Simulation from the GUI

To start a QualNet simulation from the GUI, perform the following steps.

1. Start the QualNet GUI.

2. In the **File System** panel, navigate to the QualNet configuration (.config) file for the scenario (which is equivalent to the VR-Forces scenario), right-click and select **Open**. For this demo, the configuration file is QUALNET_HOME/scenarios/vrf/CommsDemo/CommsDemo-dis.config.

3. Click on the **Run Simulation** button to begin simulation initialization. Status messages are displayed in the **Output Window** panel.

4. After the scenario has finished initializing, click the **Play** button to start the simulation.

2.3.5.4 Running the Joint Simulation Exercise

To simulate the scenario jointly in VR-Forces and QualNet, press the **Play** button in VR-Forces. Visualization of the scenario should progress in both VR-Forces and QualNet.

Expected Behavior

In this scenario, one of the tanks tells the UAV to turn on its sensors and look for the enemy. The UAV then moves towards the enemy and reports their location to the tanks. The two tanks then move towards the enemy and destroy them.

Re-running the Simulation

To run the simulation again, do the following:

1. If the simulation is still running, stop it by pressing the **Stop** button in the QualNet GUI or pressing **Ctrl+C** in the command line window.

2. In VR-Forces, press the **Pause** button and then the **Rewind** button.

3. Restart the simulation in QualNet, as described in .

4. In VR-Forces, press the **Play** button.

3 Socket Interface

This chapter describes the details of the Socket Interface and is organized as follows:

- Section 3.1 describes the messages exchanged between QualNet and an external program using the Socket Interface.
- Section 3.2 describes some features of the Socket Interface.
- Section 3.3 describes the configuration parameters required for the Socket Interface.
- Section 3.4 describes the output files produced by running the Socket Interface.

> **Note:** The Socket Interface module is included in the base QualNet installation. You do not need to install any other component or to compile QualNet in order to use the Socket Interface. However, to use the Socket Interface, you will need a license for the Federation Interfaces Library. Also, to access the pre-configured scenarios for the Socket Interface, you must install the Federation Interfaces Library (see Section 2.1.2 for details). The scenarios are installed in QUALNET_HOME/scenarios/socket-interface.

3.1 Socket Interface Messages

This section describes the inter-process communication between QualNet and an external program using the Socket Interface.

Communication between QualNet and the external program is implemented over a TCP socket, with QualNet acting as the server and the external program as the client. Several types of messages can be sent between the two processes.

Figure 3-1 shows the different QualNet states and transitions that occur when interface messages are received.
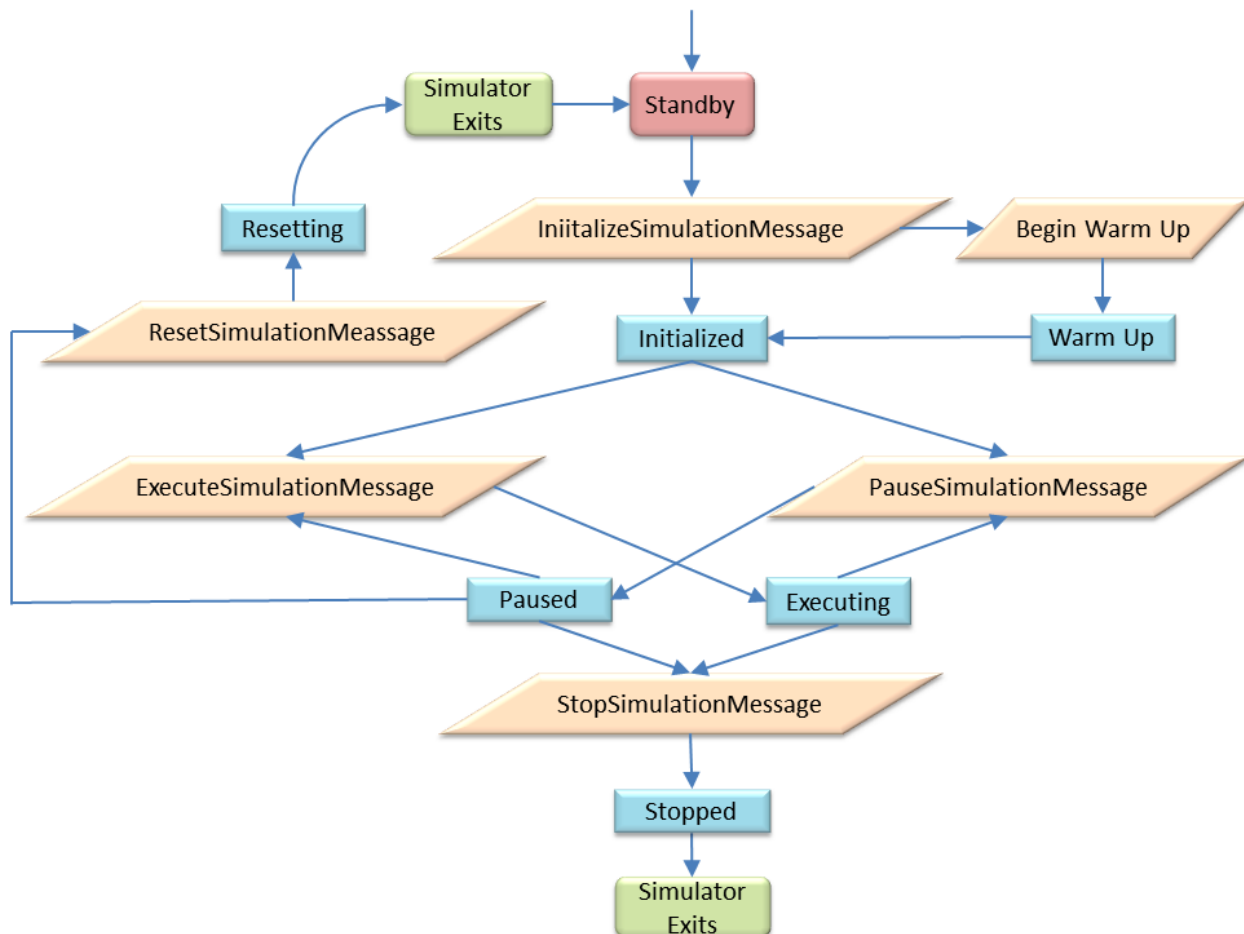
FIGURE 3-1.   Socket State Transition Diagram

Section 3.1.1 describes the structure of interface messages. Section 3.1.2 describes the request messages, i.e., messages sent from the external program to QualNet. Section 3.1.3 describes the response messages, i.e., messages sent from QualNet to the external program.

3.1.1 Structure of Interface Messages

3.1.1.1 Data Types Used in Interface Messages

The size and data encoding for data types used in the interface messages are indicated in Table 3-1.

TABLE 3-1.   Size of Data Types

Data Type	Size (in bytes)
Int8, UInt8	1
Int16, UInt16	2
Int32, UInt32	4
Int64, UInt64	8

TABLE 3-1.   Size of Data Types (Continued)

Data Type	Size (in bytes)
Float64	8
Time	8
Coordinate	24
ListofUInt8	1 * length of the list
String	variable, encoded as 2 bytes containing the length of the string followed by the string with no terminating 0 value
LongString	variable, encoded as UInt32 (4 bytes) containing the length of the string followed by the string with no terminating 0 value
ListofString	variable, encoded as consecutive strings. The serialization and deserialization of this data type depends on the message context.
ListofGroups	variable, encoded as consecutive strings. This data type is used to encode a list of multicast groups. A multicast group is an IP address in the range 224.0.0.0 to 239.255.255.255. The serialization and deserialization of this data type is explained below.

Serialization of ListofGroups Data Type

The ListofGroups data type is serialized as consecutive strings. There is no explicit count for the number of strings in the field because the number of strings can be deduced from size of the message field. Each string in the ListofGroups is preceded by a 2-byte field that denotes the length of the string.

For example, the following ListofGroups (which is 44 bytes long) encodes three strings of lengths 13, 14, and 11 bytes:

```
<LEN-1><STRING-1><LEN-2><STRING-2><LEN-3><STRING-3>
```

where

`<LEN-1>`	Length of string 1 (13). This field is 2 bytes long.
`<STRING-1>`	String 1. This is 13 bytes long.
`<LEN-2>`	Length of string 2 (14). This field is 2 bytes long.
`<STRING-2>`	String 2. This is 14 bytes long.
`<LEN-3>`	Length of string 3 (11). This field is 2 bytes long.
`<STRING-3>`	String 3. This is 11 bytes long.

3.1.1.2 Format of Interface Messages

The messages exchanged between QualNet and the external program via the Socket Interface have the following format:

> `<Message Header> <Required Fields> <Option 1> ... <Option n>`

The `Message Header` is made up of four fields, which are described in Table 3-2.

TABLE 3-2.   Fields of a Message Header

Field	Data Type	Description
MessageType	UInt8	A unique integer identifying the message type. The `MessageType` of each message is listed with the description of the message in the following subsections.
NumOptionFields	UInt8	Number of optional fields in the message.
Reserved	UInt16	Reserved for future use. This field should be set to 0.
MessageSize	UInt32	Total message size, including the message header, in bytes.

The `Required Fields` of each message are described in the following subsections.

Each `Option` has the following format:

> `<Option Header> <Option Fields>`

The `Option Header` is made up of four fields, which are described in Table 3-3.

TABLE 3-3.   Fields of an Option Header

Field	Data Type	Description
OptionType	UInt8	A unique integer identifying the option type. The `OptionType` of each option is listed with the description of messages in the following subsections.
Reserved	UInt8	Reserved for future use. This field should be set to 0.
Reserved	UInt16	Reserved for future use. This field should be set to 0.
OptionSize	UInt32	Total option size, including the option header, in bytes.

The `Option Fields` for each message are described in the following subsections.

3.1.2 Request Messages

This section describes the messages sent from the external program to QualNet via the Socket Interface.

3.1.2.1 `InitializeSimulation` Message

This message is sent from the external program to QualNet to start a QualNet simulation. QualNet does not begin the simulation until an `InitializeSimulation` message is received. This message is required for both time managed and real time modes.

QualNet moves to the Initialized state after this message is processed.

Message Type: The `MessageType` field of the `InitializeSimulation` message is set to 1.

Required Fields: The required fields of the `InitializeSimulation` message are described in Table 3-4.

TABLE 3-4.   **Required Fields of `InitializeSimulation` Message**

Field	Data Type	Description

TimeManagementMode	UInt8	Time management mode.
		
		    0: Time managed mode
		
		    1: Real time mode

Options: The optional fields of the `InitializeSimulation` message and the corresponding `OptionType` are described in Table 3-5.

TABLE 3-5.   **Optional Fields of `InitializeSimulation` Message**

OptionType	Option Field	Data Type	Description

0	CoordinateSystem	UInt8	Coordinate system.
			
			    0: Cartesian (default)
			
			    1: Lat/Lon/Alt
			
			    2: GCC Cartesian

1	Scenario	LongString	Description of the simulation scenario.

2	SourceResponseMulticast	Bool	Indication whether responses to multicast messages should be sent to the message source.
			
			    YES: Responses to multicast messages should be sent to the source of the message.
			
			    NO: Responses to multicast messages should be sent to the multicast group owner (default).

3.1.2.2 `PauseSimulation` Message
This message is sent from the external program to QualNet to move QualNet to the Paused state from the Initialized state or Executing state.

Message Type: The `MessageType` field of the `PauseSimulation` message is set to 2.

Required Fields: This message has no required fields.

Options: The optional fields of the `PauseSimulation` message and the corresponding `OptionType` are described in Table 3-6.

TABLE 3-6.   Optional Fields of PauseSimulation Message

OptionType	Option Field	Data Type	Description
2	PauseTime	Time	Time to pause the simulation. If this field is not included in the message or it timestamps are not enabled, QualNet will enter the pause state as soon as the message is received.

3.1.2.3 `ExecuteSimulation` Message

This message is sent from the external program to QualNet to move QualNet to the Execute state from the Pause or Initialized state.

Message Type: The `MessageType` field of the `ExecuteSimulation` message is set to 3.

Required Fields: This message has no required fields.

Options: This message has no optional fields.

3.1.2.4 `StopSimulation` Message

This message is sent from the external program to QualNet to gracefully end a QualNet simulation. When this message is received, QualNet statistics are printed to various files (see Section 3.4), socket connections are closed, and QualNet moves to the Stopping state.

Message Type: The `MessageType` field of the `StopSimulation` message is set to 4.

Required Fields: This message has no required fields.

Options: The optional fields of the `StopSimulation` message and the corresponding `OptionType` are described in Table 3-7.

TABLE 3-7.   Optional Fields of `StopSimulation` Message

OptionType	Option Field	Data Type	Description
2	StopTime	Time	Time to stop the simulation. If this field is not included in the message or if timestamps are not enabled, QualNet ends as soon as the message is received.

3.1.2.5 `ResetSimulation` Message

This message is sent from the external program to QualNet to clean QualNet internal data structures and return QualNet to the Standby state. This gracefully ends the QualNet simulation like the `StopSimulation` message. QualNet socket connections are closed and statistics are output to various files (see Section 3.4). Following that, QualNet enters its original state and waits for socket connections to begin a new scenario.

QualNet moves to the Resetting state before it enters the Standby state.

Message Type: The `MessageType` field of the `ResetSimulation` message is set to 5.

Required Fields: This message has no required fields.

Options: The optional fields of the `ResetSimulation` message and the corresponding `OptionType` are described in Table 3-8.

TABLE 3-8.   Optional Fields of `ResetSimulation` Message

OptionType	Option Field	Data Type	Description
2	ResetTime	Time	Time to reset the simulation. If this field is not included in the message or if timestamps are not enabled, QualNet resets as soon as the message is received. Parallel execution of the message is not supported.

3.1.2.6 `AdvanceTime` Message

This message is sent from the external program to QualNet to advances the simulation time when running in time managed mode.

Message Type: The `MessageType` field of the `AdvanceTime` message is set to 6.

Required Fields: The required fields of the `AdvanceTime` message are described in Table 3-9.

TABLE 3-9.   Required Fields of `AdvanceTime` Message

Field	Data Type	Description
TimeAllowance	Time	Maximum allowable simulation time.

Options: This message has no optional fields.

3.1.2.7 `DynamicCommand` Message

This message is sent from the external program to QualNet to query or modify scenario parameters during the simulation.

Message Type: The `MessageType` field of the `DynamicCommand` message is set to 8.

Required Fields: The required fields of the `DynamicCommand` message are described in Table 3-10.

.

TABLE 3-10.   Required Fields of `DynamicCommand` Message

Field	Data Type	Description
Type	UInt8	Type of operation to perform. 0: Read 1: Write 2: Execute

TABLE 3-10.   Required Fields of **DynamicCommand** Message (Continued)

Field	Data Type	Description
Path	String	Path of dynamic object to perform operation on.
Args	String	Arguments to write or execute. This field is ignored for read operations.

Options: This message has no optional fields.

3.1.2.8 **CreatePlatform** Message

This message is sent from the external program to QualNet to create a new platform in the simulation.

Message Type: The MessageType field of the CreatePlatform message is set to 10.

Required Fields: The required fields of the CreatePlatform message are described in Table 3-11.

TABLE 3-11.   Required Fields of **CreatePlatform** Message

Field	Data Type	Description
EntityId	String	Identification of the entity to create.
Position	Coordinate	Initial position of the entity.
State	UInt8	Initial damage state of the entity.

Options: The optional fields of the CreatePlatform message and the corresponding OptionType are described in Table 3-12.

TABLE 3-12.   Optional Fields of **CreatePlatform** Message

OptionType	Option Field	Data Type	Description
2	CreateTime	Time	Time when the platform is created. If this field is not included in the message, the platform is created as soon as the message is received.
4	Type	UInt8,	Platform type. 0: Ground (default) 1: Air
5	MulticastGroups	ListofGroups	List of multicast groups that this platform is to join. A multicast group is represented by an IP address in the range 224.0.1.0 to 239.255.255.255.
22	Velocity	Coordinate	Vector with three elements. If coordinate system is Cartesian, velocity is (X/sec, Y/sec, Z/sec). If coordinate system is Lat-Lon-Alt, velocity is (Lat/sec, Lon/sec, Alt/sec). If coordinate system is GCC Cartesian, velocity is (X/sec, Y/sec, Z/sec).

3.1.2.9 `UpdatePlatform` Message

This message is sent from the external program to QualNet to update the position and/or the state of a platform.

Message Type: The `MessageType` field of the `UpdatePlatform` message is set to 11.

Required Fields: The required fields of the `UpdatePlatform` message are described in Table 3-13.

TABLE 3-13.   Required Fields of `UpdatePlatform` Message

Field	Data Type	Description
EntityId	String	Identification of the entity to update or IP address for which only multicast groups will be modified.

Options: The optional fields of the `UpdatePlatform` message and the corresponding `OptionType` are described in Table 3-14.

TABLE 3-14.   Optional Fields of `UpdatePlatform` Message

OptionType	Option Field	Data Type	Description
2	UpdateTime	Time	Time to update the platform.
			If this field is not included in the message, the platform is updated as soon as the message is received.
6	Position	Coordinate	New position.
			If this field is not included in the message, the platform's position is not changed.
7	State	UInt8	New state.
			0: Communications are not damaged
			1: Communications are damaged
			If this field is not included in the message, the platform's state is not changed.
9	JoinMulticastGroups	ListofGroups	List of multicast groups to join.
10	LeaveMulticastGroups	ListofGroups	List of multicast groups to leave.
22	Velocity	Coordinate	Vector with three elements.
			If coordinate system is Cartesian, velocity is (X/sec, Y/sec, Z/sec).
			If coordinate system is Lat-Lon-Alt, velocity is (Lat/sec, Lon/sec, Alt/sec).
			If coordinate system is GCC Cartesian, velocity is (X/sec, Y/sec, Z/sec).

3.1.2.10 `CommEffectsRequest` Message

This message is sent from the external program to QualNet and models communications between two nodes. A `CommEffectsResponse` is sent to the external program indicating whether the communication was successful or unsuccessful, along with the communication latency experienced. The fields `Id1` and `Id2` provide 16 bytes for message identification. `Id1` and `Id2` are returned to the external program in the `CommEffectsResponse` message. Failure messages are not sent for multicast or broadcast traffic.

Message Type: The `MessageType` field of the `CommEffectsRequest` message is set to 12.

Required Fields: The required fields of the `CommEffectsRequest` message are described in Table 3-15.

TABLE 3-15.   Required Fields of `CommEffectsRequest` Message

Field	Data Type	Description
Id1	UInt64	First 8 bytes of message identification field.
Id2	UInt64	Second 8 bytes of message identification field.
Protocol	UInt8	Protocol for this message. 0: TCP 1: UDP 2: Network
Size	UInt32	Size of the simulated packet, in bytes.
SenderId	String	Entity ID of the message sender.
ReceiverId	String	Entity ID of the message receiver or multicast group. `255.255.255.255` is used for a broadcast message.

Options: The optional fields of the `CommEffectsRequest` message and the corresponding `OptionType` are described in Table 3-16.

TABLE 3-16.   Optional Fields of `CommEffectsRequest` Message

OptionType	Option Field	Data Type	Description
2	SendTime	Time	Time to send the message. If this field is not included in the message, the request is sent as soon as the message is received.
11	Precedence	UInt8	Message priority. 0: Routine (default) 1: Priority 2: Immediate 3: Flash 4: Flash Override 5: Critical 6: Internet Control 7: Net Control **Note:** A message can not include more than one of the three options: `Precedence`, `DSCP`, and `TOS`. If none of `Precedence`, `DSCP`, and `TOS` is specified, then Routine priority is used.
12	Description	String	Description of this request. This is echoed back in the `CommEffectsResponse` message sent in response to this message.

TABLE 3-16.   Optional Fields of `CommEffectsRequest` Message (Continued)

OptionType	Option Field	Data Type	Description
13	FailureTimeout	Time	Message failure timeout value (in seconds). If this field is not included in the message, then the timeout value specified in the configuration file (SOCKET-INTERFACE-UDP-FAILURE-TIMEOUT or SOCKET-INTERFACE-TCP-FAILURE-TIMEOUT) is used.
18	DSCP	UInt8	Request DSCP. The valid range for this field is 0-63. The bits indicate the following: 0-2: Precedence 3: Low Delay 4: High Throughput 5: High Reliability For example, if the DSCP field is set to 10 (001010 in binary), then the message is given a precedence value of 1 with High Throughput. **Note:** A message can not include more than one of the three options: Precedence, DSCP, and TOS.
19	TOS	UInt8	Request TOS. The valid range for this field is 0-255 (i.e., all 8 bits can be used). This value replaces the entire TOS byte in the IP header. **Note:** A message cannot include more than one of the three options: Precedence, DSCP, and TOS.
24	TTL	UInt8	Time-To-Live (TTL) field for communications. This is valid for TCP and UDP communications. For TCP communications, the first TCP packet sent from the source to the destination should specify the TTL that should be used for all subsequent packets from the source to the destination.

3.1.2.11 `GetRequest` Message

This message is sent from the external program to QualNet to query the Management Information Base (MIB) variable. This message allows the user to retrieve the value of a list of MIB variables.

Message Type: The `MessageType` field of the `GetRequest` message is set to 15.

Required Fields: The required fields of the GetRequest message are described in Table 3-17.

TABLE 3-17.   Required Fields of GetRequest Message

Field	Data Type	Description
EntityId	String	Identification of the entity on which to perform the request operation.
NumOIDs	UInt16	Number of OIDs.
OID	ListofString	List of object identifiers on which to perform the operation. This list should have a number of strings equal to NumOIDs.

Options: This message has no optional fields.

3.1.2.12 SetRequest Message

This message is sent from the external program to QualNet to set the value of a MIBS variable.

Message Type: The MessageType field of the SetRequest message is set to 17.

Required Fields: The required fields of the SetRequest message are described in Table 3-18.

TABLE 3-18.   Required Fields of SetRequest Message

Field	Data Type	Description
EntityId	String	Identification of the entity on which to perform the request operation.
NumOIDs	UInt16	Number of OIDs.
OID	ListofString	List of object identifiers on which to perform the operation. This list should have a number of strings equal to NumOIDs.
Values	ListofString	New values to set each OID to. The number of strings should be equal to NumOIDs. Each value corresponds to the OID field with the matching index.

Options: This message has no optional fields.

3.1.2.13 GetNextRequest Message

This message is sent from the external program to QualNet to query the Management Information Base (MIB) variable. For each specified OID, QualNet will return its lexicographical successor in a GetResponse message. For example, the lexicographical successor of 1.3.6.1.2.1.1.3 is 1.3.6.1.2.1.1.3.0, and the lexicographical successor of 1.3.6.1.2.1.1.3.0 is 1.3.6.1.2.1.1.5.0.

Message Type: The MessageType field of the GetNextRequest message is set to 18.

Required Fields: The required fields of the `GetNextRequest` message are described in Table 3-19.

TABLE 3-19.   Required Fields of `GetNextRequest` Message

Field	Data Type	Description
EntityId	String	Identification of the entity on which to perform the request operation.
NumOIDs	UInt16	Number of OIDs.
OID	ListofString	List of object identifiers on which to perform the operation. The number of strings should be equal to `NumOIDs`.

Options: This message has no optional fields.

3.1.2.14 `GetBulkRequest` Message

This message is sent from the external program to QualNet to query the Management Information Base (MIB) variable. This message allows the user to retrieve a large section of the MIB. QualNet will perform one GetNext operation on each of the first `NumNonRepeat` OIDs specified, followed by a maximum of `MaxRepeat` GetNext operations on each of the remaining OIDs. This means that the maximum number of objects returned is `NumNonRepeat` + (`NumOIDs` - `MaxNumNonRepeat`) * `MaxRepeat`.

Message Type: The `MessageType` field of the `GetBulkRequest` message is set to 19.

Required Fields: The required fields of the `GetBulkRequest` message are described in Table 3-20.

TABLE 3-20.   Required Fields of `GetBulkRequest` Message

Field	Data Type	Description
EntityId	String	Identification of the entity on which to perform the request operation.
NumOIDs	UInt16	Number of OIDs.
NumNonRepeat	UInt16	Number of scalar MIB objects.
MaxRepeat	UInt16	Maximum repetition of a non-scalar MIB object.
MibsId	ListofString	List of object identifiers on which to perform the operation. The number of strings should be equal to `NumOIDs`.

Options: This message has no optional fields.

3.1.2.15 `QuerySimulationState` Message

This message is sent from the external program to QualNet to query the simulation state of QualNet. A `SimulationState` message is sent to the external program with the current state of QualNet.

Message Type: The `MessageType` field of the `QuerySimulationState` message is set to 20.

Required Fields: This message has no required fields.

Options: This message has no optional fields.

3.1.2.16 `BeginWarmup` Message

This message is sent from the external program to QualNet to start the warm up phase (see Section 3.2.3). This message is sent after the `InitializeSimualtion` and `CreatePlatform` messages are sent to QualNet.

Message Type: The `MessageType` field of the `BeginWarmup` message is set to 21.

Required Fields: This message has no required fields.

Options: This message has no optional fields.

3.1.3 Response Messages

This section describes the messages sent from QualNet to the external program.

3.1.3.1 `SimulationState` Message

This message is sent from QualNet to the external program when QualNet changes state. Every new connection from an external program receives a `SimulationState` message containing the current QualNet state.

QualNet begins in the Standby state.

Message Type: The `MessageType` field of the `SimulationState` message is set to 0.

Required Fields: The required fields of the `SimulationState` message are described in Table 3-21.

TABLE 3-21.   Required Fields of `SimulationState` Message

Field	Data Type	Description
State	UInt8	Current state. 1: Standby 2: Initialized 3: Paused 4: Executing 6: Resetting 8: Stopping 10: Warmup
OldState	UInt8	Previous state.

Options: This message has no optional fields.

3.1.3.2 `SimulationIdle` Message

This message is sent from QualNet to the external program when QualNet is idling, i.e., waiting for a time advance from the external program. This message is only used when QualNet is running in time managed mode. QualNet resumes the simulation when its time is advanced by an `AdvanceTime` message.

Message Type: The `MessageType` field of the `SimulationIdle` message is set to 7.

Required Fields: The required fields of the `SimulationIdle` message are described in Table 3-22.

TABLE 3-22.   Required Fields of `SimulationIdle` Message

Field	Data Type	Description
CurrentTime	Time	Simulation time when QualNet starts idling, waiting for a time advance from the external program.

Options: This message has no optional fields.

3.1.3.3 `DynamicResponse` Message

This message is sent from QualNet to the external program to provide the result of the operation requested by a `DynamicCommand` message.

Message Type: The `MessageType` field of the `DynamicResponse` message is set to 9.

Required Fields: The required fields of the `DynamicResponse` message are described in Table 3-23.

TABLE 3-23.   Required Fields of `DynamicResponse` Message

Field	Data Type	Description
Type	UInt8	Type of operation that was performed. 0: Read 1: Write 2: Execute
Path	String	Path of dynamic object that operation was performed on.
Args	String	Arguments passed in the originating DynamicCommand.
Output	String	Output of dynamic command.

Options: This message has no optional fields.

3.1.3.4 `CommEffectsResponse` Message

This message is sent from QualNet to the external program and provides a response to a `CommEffectsRequest` message indicating the result of the message delivery request.

Message Type: The `MessageType` field of the `CommEffectsResponse` message is set to 13.

Required Fields: The required fields of the `CommEffectsResponse` message are described in Table 3-24

TABLE 3-24.   Required Fields of `CommEffectsResponse` Message

Field	Data Type	Description
Id1	UInt64	First 8 bytes of message identification field.
Id2	UInt64	Second 8 bytes of message identification field.
SenderId	String	Entity identification of the message sender.
ReceiverId	String	Entity identification of the message receiver.

TABLE 3-24.   Required Fields of `CommEffectsResponse` Message (Continued)

Field	Data Type	Description
Status	Int8	Indication of success or failure. 0: Success 1: Failure due to expired FailureTimeout window. Any nonzero value indicates a failure.
ReceiveTime	Time	Time when the message was received. This will be the `SendTime` field of the `CommEffectsRequest` message plus the `Latency`. field of the `CommEffectsResponse` message. In real time mode, the simulation time at which the message was sent is used as `SendTime`.
Latency	Time	Time spent by the message in transit (`ReceiveTime` - `SendTime` field of the `CommEffectsRequest` message).

Options: The optional fields of the `CommEffectsResponse` message and the corresponding `OptionType` are described in Table 3-25.

TABLE 3-25.   Optional Fields of `CommEffectsResponse` Message

OptionType	Option Field	Data Type	Description
12	Description	String	Description of the request. This is echoed back from the `CommEffectsRequest` message in response to which this message is sent.

3.1.3.5 `Error` Message

This message is sent from QualNet to the external program when an incorrect message is received. This message contains a numeric error code, a human-readable error string, and the entire original message that generated the error.

Message Type: The `MessageType` field of the `Error` message is set to 14.

Required Fields: The required fields of the `Error` message are described in Table 3-26.

.

TABLE 3-26.   Required Fields of `Error` Message

Field	Data Type	Description
Code	UInt8	Error code.
		0: No error
		1: Configuration Error
		2: Message Creation Error
		3: Invalid Message
		4: Invalid Transition
		5: Invalid EntityId
		6: Invalid State
		7: Invalid Coordinates
		8: Invalid ReceiverId
		9: Empty Receiver List
		10: Invalid Senderid
		11: Invalid Send Time
		12: Invalid Stop Time
		13: Invalid SocketId
		14: Simulator Warning
		15: Simulator Error
		16: Socket Error
		17: Invalid Operation Type
		18: Empty Scenario String
		19: Invalid Dynamic Command
		20: Invalid Protocol
		21: Invalid Group
		28: Invalid Simulation State
		29: Invalid Node ID
Error	String	Human-readable error string.

Options: The optional fields of the `Error` message and the corresponding `OptionType` are described in Table 3-27.

TABLE 3-27.   Optional Fields of `Error` Message

OptionType	Option Field	Data Type	Description
16	OriginatingMessage	QualNet Message	If a message caused the error, this field contains the complete error-causing message as it was received on the socket.

3.1.3.6 `GetResponse` Message

This message is sent from QualNet to the external program in response to a SNMP command (i.,e., `GetRequest`, `GetNextRequest`, `GetBulkRequest`, and `SetRequest` messages).

Message Type: The `MessageType` field of the `GetResponse` message is set to 16.

Required Fields: The required fields of the `GetResponse` message are described in Table 3-28

TABLE 3-28.   Required Fields of `GetResponse` Message

Field	Data Type	Description
EntityId	String	Identification of the entity on which the request operation is performed.
NumOIDs	UInt16	Number of OIDs.
OID	ListofString	List of object identifiers on which the MIBS object operations were performed. The number of strings is equal to `NumOIDs`.
Output	ListofString	List of the outputs from the operation performed on the MIBS objects. The number of strings is equal to `NumOIDs`. Each output value corresponds to the `OID` field with the matching index.
ErrorStatus	ListofUInt8	List of SNMP errors. The following codes are used. 0: `noError`: No error. 1: `tooBig`: Message size is too big. 2: `noSuchName`: OID does not exist. 3: `badValue`: `Values` field of the `SetRequest` message is incorrect. 4: `readOnly`: Object was set by a `SetRequest` message but is read only. 5: `genErr`: Any other error. 6: `endOfMibView`: `GetNextRequest` or `GetBulkRequest` message has reached the last object in the database The number of errors is equal to `NumOIDs`. Each error value corresponds to the OID field with the matching index. Currently `tooBig`, `badValue`, and `genErr` are not returned.

Options: This message has no optional fields.

3.2 Features of Socket Interface

This section describes some features of the Socket Interface.

3.2.1 Multicast and Broadcast Support

Platforms are added to multicast groups by the `MulticastGroups` field in the `CreatePlatform` message and the `JoinMulticastGroups` field in the `UpdatePlatform` message. Platforms are

removed from multicast groups by the `LeaveMulticastGroups` field in the `UpdatePlatform` message. Multicast routing must be configured in the QualNet scenario configuration file.

Multicast and broadcast responses are handled differently from unicast responses. Unicast `CommEffectsResponse` messages are always sent back to the external program that initiated the `CommEffectsRequest` message. For multicast and broadcast `CommEffectsRequest` messages, the external program that receives the `CommEffectsResponse` may be different from the external program that initiated the `CommEffectsRequest`. QualNet handles `CommEffectsResponse` messages for multicast and broadcast as follows:

1. QualNet checks if an external program has added the platform to the multicast group or broadcast address. If so, QualNet sends a `CommEffectsResponse` with the `ReceiverId` set to the receiving platform's entity identifier to all external programs that have added the platform to the multicast group.

   The originator of the `CommEffectsRequest` message receives the response only if it has added the platform to the multicast group.

2. If no external program has added the platform to the multicast group or broadcast address, QualNet sends the `CommEffectsResponse` to the originator of the `CommEffectsRequest` message.

Multiple external programs may initiate a request to add the same platform to the same multicast group. QualNet adds the platform to the multicast group only once. However, each external program that initiates such a request receives a copy of the `CommEffectsResponse` for the given platform and multicast group. An external program may add a platform to a broadcast address in which case it will receive responses for the broadcast address.

When an external program sends a request to remove a platform from a multicast group, the platform is removed from the group even if other external programs have added that platform to the same multicast group.

3.2.2 Dynamic Commands

Dynamic commands are supported by the Socket Interface to allow an external program to view the QualNet scenario state and make modifications to the QualNet scenario while it is running. For example, using dynamic commands, a user can check if a platform is a subnet gateway and, if it not a gateway, the user can turn it into a subnet gateway.

The parts of the QualNet simulation that can be changed dynamically are organized in a directory structure called the Dynamic Hierarchy. The following is an example of a portion of a dynamic hierarchy:

```
/platform/A100/interface/192.0.0.2/PHY-ABSTRACT-DATA-RATE
/platform/A132/interface/192.0.0.3/PHY-ABSTRACT-DATA-RATE
/platform/A132/interface/192.0.3.101/PHY-ABSTRACT-DATA-RATE
/platform/B233/interface/192.0.3.122/PHY-ABSTRACT-DATA-RATE
```

This example shows three platforms: A100, A132 and B233. A100 and B233 belong to subnets 192.0.0.0 and 192.0.3.0 respectively. A132 belongs to both subnets because it has interfaces to both 192.0.0.0 and 192.0.3.0 subnets.

The variable `PHY-ABSTRACT-DATA-RATE` can be read and modified at any point in the simulation via a `DynamicCommand` message (see Section 3.1.2.7). For example, to read the value of the `PHY-`

`ABSTRACT-DATA-RATE` variable for platform A100, a `DynamicCommand` message with the following fields can be used:

Field	Value	Description
Type	0	Indication of a read operation.
Path	"/platform/A100/interface/192.0.0.2/PHY-ABSTRACT-DATA-RATE"	Path of dynamic object to perform the read operation on.
Args	""	Empty string.

The results of this operation are sent back in a `DynamicResponse` message (see Section 3.1.2.8), which may have the following fields:

Field	Value	Description
Type	0	Indication that a read operation was performed.
Path	"/platform/A100/interface/192.0.0.2/PHY-ABSTRACT-DATA-RATE"	Path of dynamic object on which the read operation was performed.
Args	""	Empty string.
Output	"2000000000"	Output of the dynamic command.

To change the value of the `PHY-ABSTRACT-DATA-RATE` variable for platform B233, a `DynamicCommand` message with the following fields can be used:

Field	Value	Description
Type	1	Indication of a write operation.
Path	"platform/B233/interface/192.0.0.2/PHY-ABSTRACT-DATA-RATE"	Path of dynamic object to perform the write operation on.
Args	"2500000000"	New value of the variable.

This will change the PHY Abstract data rate for platform B233. No `DynamicResponse` message is sent for write operations.

Table 3-29 summarizes the dynamic variables that are available for all platforms created through the Socket Interface. The interface address can be determined from the configuration file by tracing the

platform mapping back to the node identifier using the entity mapping file (see Section 3.3.1.1). It can also be determined from the output file graph.log (see Section 3.4.3).

TABLE 3-29.   Available Dynamic Variables and Results of Dynamic Commands

Path	Result of Read	Result of Write	Result of Execute
`/platform/<entityID>/interface/` `<ipAddress>/PHY-ABSTRACT-DATA-RATE`	Returns the current data rate (in bits per second).	Changes the data rate to the specified value (in bits per second).	Not applicable.
`/platform/<entityID>/interface/` `<ipAddress>/fault`	• Returns "yes" if the interface is faulted. • Returns "no" if the interface is not faulted. (Interface faults by dynamic commands are independent of a platform's damage state.)	Not applicable.	• Creates an interface fault if "yes" is specified • Interface is no longer faulted if "no" is specified.
`/platform/<entityID>/` `multicastGroups`	Returns list of multicast groups the platform is a member of, delimited by spaces.	Not applicable.	Not applicable.

3.2.3 Warm-up Phase

QualNet supports an optional phase that occurs between the Standby and Initialize phases called the Warm-up phase (see Figure 3-1). The warm-up phase provides additional time before the beginning of the simulation for the routing protocols to converge.

QualNet operation in warm-up phase works as follows:

1. QualNet loads the scenario configuration file.
2. QualNet enters the Standby state.

    Note: CommEffectsRequest messages received in the Standby state are treated as errors.
3. QualNet receives an InitializeSimulation message.
4. If a warm-up time is specified in the configuration file, then QualNet waits to receive the BeginWarmup message.

    a. Upon receiving the BeginWarmup message, QualNet enters the WarmUp state and begins the warm-up phase. In the warm-up phase, QualNet operates as follows:

    • QualNet processes CreatePlatform, UpdatePlatform, and CommEffectsRequest messages. CommEffectsRequest messages received in this phase are either processed as successes with 0 delay or dropped, depending on the warm-up parameters (see Section 3.2.3.1).
    • QualNet buffers unused messages until they are needed.

    b. QualNet begins running the scenario, forming routing tables, etc.

    c. QualNet processes statistics database queries during the warm-up phase. Database timestamps during the warm-up phase are negative. For example, if the warm-up phase duration is 10 minutes, then database timestamps will count from -10 minutes up to 0 minutes (warm-up time over).

    d. QualNet finishes warm-up phase and enters the Initialized state.

5. QualNet enters the Initialized state. If QualNet enters the Initialized state from the WarmUp state (i.e., if a warm-up time is specified in the configuration file), then QualNet responds to `CommEffectsRequest` messages received in the Initialized state with success and 0 delay.

6. QualNet receives a `PauseSimulation` message and enters the Paused state.

7. QualNet receives an `ExecuteSimulation` message and enters the Executing state. At this point, QualNet's internal simulation time is equal to the warm-up time. However, when communicating with external software QualNet will consider real time and simulation time to be zero.

8. QualNet runs as normal.

> **Note:** The warm-up time is subtracted from all timestamps sent to the external program. The warm-up time is added to all timestamps received from the external program. This makes it appear that QualNet is at time 0 when transitioning to the execute phase.

3.2.3.1 Configuring Warm-up Phase Parameters

This section describes how to configure the warm-up phase parameters

3.2.3.1.1 Command Line Configuration

To configure the warm-up phase parameters for the command line interface, include the parameters listed in Table 3-30 in the scenario configuration (.config) file.

TABLE 3-30.   Warm-up Phase Parameters

Parameter	Value	Description
`EXTERNAL-WARM-UP-TIME` *Optional* *Scope:* Global	Time *Range:* ≥ `0S` *Default:* `0S`	Length of the warm-up phase.
`EXTERNAL-WARM-UP-DROP` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether packets received from external sources in the warm-up phase are dropped or delivered to the destination with zero delay. By default external packets are delivered to the destination with zero delay.

3.2.3.1.2 GUI Configuration

To configure the warm-up phase parameters in the GUI, perform the following steps:

1. Go to **Scenario Properties Editor > External Interfaces > Warm-up Phase**.

2. To enable the warm-up phase, set **Enable Warm-up Phase** to *Yes* and set the dependent parameters listed in Table 3-31.
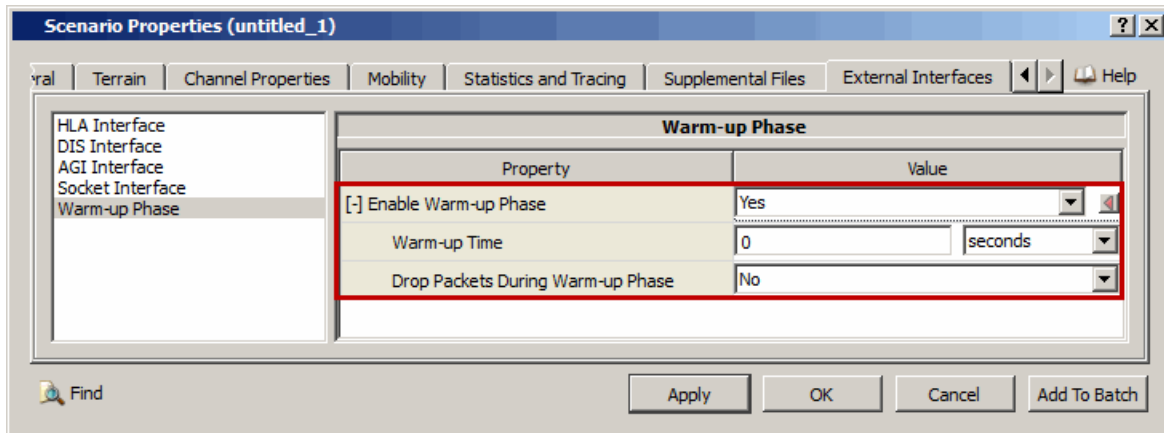
FIGURE 3-2.   Setting Warm-up Phase Parameters

TABLE 3-31.   Command Line Equivalent of Warm-up Phase Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Warm-up Time	Global	`EXTERNAL-WARM-UP-TIME`
Drop Packets During Warm-up Phase	Global	`EXTERNAL-WARM-UP-DROP`

3.3 Socket Interface Configuration

This section describes the parameters that need to be configured for the Socket Interface. These parameters configure QualNet to handle messages from the external program over the Socket Interface. The QualNet network scenario must be configured in addition to the Socket Interface.

Section 3.3.1 describes how to set up these parameters in the QualNet scenario configuration file.
Section 3.3.2 describes how to set up these parameters using the QualNet GUI.

3.3.1 Command Line Configuration

Table 3-32 lists the QualNet scenario configuration (.config) file parameters required for the Socket Interface. See Section 1.1.1 for a description of the format for specifying parameters in the QualNet configuration file and the format used for the parameter table.

TABLE 3-32. Socket Interface Configuration Parameters

Parameter	Value	Description
SOCKET-INTERFACE *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Indicates whether QualNet should use the Socket Interface. This parameter must be included and set to YES for the Socket Interface to be initialized.
SOCKET-INTERFACE-NUM-PORTS *Optional* *Scope:* Global	Integer *Range:* > 0 *Default:* 1	Number of ports socket interface is to open for incoming connections. Multiple connections may be created per port.
SOCKET-INTERFACE-PORT *Optional* if SOCKET-INTERFACE-NUM-PORTS = 1 *Required* if SOCKET-INTERFACE-NUM-PORTS > 1 *Scope:* Global *Instances:* index	Integer *Range:* > 0 *Default:* See description.	Port numbers for socket interface. If SOCKET-INTERFACE-NUM-PORTS is 1, then this parameter can be omitted (in which case the default port number is 5033) or the port number can be specified as parameter SOCKET-INTERFACE-PORT or SOCKET-INTERFACE-PORT[0]. If SOCKET-INTERFACE-NUM-PORTS is greater than 1, then the configuration file should contain SOCKET-INTERFACE-NUM-PORTS instances of SOCKET-INTERFACE-PORT[index].
SOCKET-INTERFACE-UDP-FAILURE-TIMEOUT *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 15S	Time interval to wait before sending a CommEffectsResponse message indicating a failure in response to a UDP message.
SOCKET-INTERFACE-TCP-FAILURE-TIMEOUT *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 90S	Time interval to wait before sending a CommEffectsResponse message indicating a failure in response to a TCP message.
SOCKET-INTERFACE-PRINT-PER-PACKET-STATS *Optional* *Scope:* Global	Filename or List: • STDOUT	File to print per-packet statistics for each CommEffectsRequest message. If the value is a filename, statistics are printed to that file. If the value is STDOUT, statistics are printed on the terminal window. If this parameter is not specified, per-packet statistics are not printed.
SOCKET-INTERFACE-LOG *Optional* *Scope:* Global	List: • FILE • STDOUT • NONE *Default:* FILE	File to log data. If the value is FILE, all output is logged to files (see Section 3.4). If the value is STDOUT, all output is logged to the terminal window. If the value is NONE, output is not logged.

TABLE 3-32.   Socket Interface Configuration Parameters (Continued)

Parameter	Value	Description
SOCKET-INTERFACE- PRINT-REAL-TIME *Optional* *Scope:* Global	List: • 0 • 1 *Default:* 0	Indicates whether timestamps, if printed, indicate wall-clock time or simulation time. If the value is 0, timestamps indicate simulation time. If the value is 1, timestamps indicate simulation time as well as wall-clock time.
SOCKET-INTERFACE- STATS-PRINT-REAL-TIME *Optional* *Scope:* Global	List: • 0 • 1 *Default:* 0	Indicates whether to output statistics based on real time or simulation time. If the value is 0, statistics are based on simulation time. If the value is 1, statistics are based on real time.
SOCKET-INTERFACE- STATS-PRINT-INTERVAL *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 60S	Time interval used to output statistics.
SOCKET-INTERFACE- GRAPH-PRINT-REAL-TIME *Optional* *Scope:* Global	List: • 0 • 1 *Default:* 1	Indicates whether the time interval specified by parameter SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL is real time interval or simulation time interval. If the value is 0, SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL is a simulation time interval. If the value is 1, SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL is a real time interval.
SOCKET-INTERFACE- GRAPH-PRINT-INTERVAL *Optional* *Scope:* Global	Time *Range:* ≥ 0S *Default:* 60S	Time interval used to print data to the graph log file.
SOCKET-INTERFACE-IDLE- WHEN-RESPONSE-SENT *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Indicates whether QualNet should go into idle mode when a CommEffectsResponse message is sent. If the value is YES, QualNet goes into idle mode when a CommEffectsResponse message is sent. If the value is NO, sending a CommEffectsResponse message does not cause QualNet to go into idle mode.
SOCKET-INTERFACE- ALWAYS-SUCCESS *Optional* *Scope:* Global	List: • YES • NO *Default:* NO	Indicates whether all CommEffectsResponse messages sent by QualNet should indicate a success. If the value is YES, all CommEffectsResponse messages indicate a success. If the value is NO, CommEffectsResponse messages sent in response to unsuccessful requests indicate a failure. **Note:** Multicast traffic is not affected by this parameter.

TABLE 3-32.   Socket Interface Configuration Parameters (Continued)

Parameter	Value	Description
`SOCKET-INTERFACE-CPU-HOG` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether QualNet should use up all available CPU time. If the value is `YES`, QualNet uses up all available CPU time. If the value is `NO`, QualNet does not use up all available CPU time, i.e., it uses the sleep system call.
`SOCKET-INTERFACE-ENTITY-MAPPING-FILE` *Optional* *Scope:* Global	Filename	Name of the entity mapping file. This file specifies the static mapping between the external entities and QualNet nodes. If this parameter is not specified, mapping between the external entities and QualNet nodes is done dynamically by assigning the next available QualNet node identifier to an external entity when that entity is created. The format of the entity mapping file is described in Section 3.3.1.1.
`SOCKET-INTERFACE-LOG-AUTOMATIC-FLUSH` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `YES`	Indicates whether QualNet should flush log files for each line of output. If the value is `YES`, QualNet will flush log files for each line of output. If the value is `NO`, QualNet allows the operating system to determine when log files should be flushed. This may result in a performance increase for large scenarios but individual lines of a log file will experience a delay before being committed to disk.
`SOCKET-INTERFACE-DISTRIBUTED-ENVIRONMENT` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether QualNet should support operation in a geographically distributed environment (where QualNet and its connected external programs are distributed physically and communication delays are larger than if they were in the same building). If the value is `YES`, QualNet will support operation in a geographically distributed environment. If the value is `NO`, QualNet will not provide special support for operation in a geographically distributed environment.
`SOCKET-INTERFACE-PAUSE-REPLY-ZERO-DELAY` *Optional* *Scope:* Global	List: • `YES` • `NO` *Default:* `NO`	Indicates whether QualNet should respond with 0 second delay after receiving a CommEffectsRequest message when it is in the "Pause" state. If the value is `YES`, QualNet will respond with 0 second delay for after receiving a CommEffectsRequest message when it is in the "Pause" state. If the value is `NO`, QualNet will not respond with 0 second delay for after receiving a CommEffectsRequest message when it is in the "Pause" state, and instead would buffer the messages to be processed when in "Execution" state.

TABLE 3-32.   Socket Interface Configuration Parameters (Continued)

Parameter	Value	Description

`SOCKET-INTERFACE-`
`PAUSE-ADVANCE-`
`SIMULATION-TIME`
 	
Optional
 	
Scope: Global
	List:
	• `YES`
	• `NO`
	
	Default: `NO`
		Indicates whether QualNet should advance the simulation time in "Pause" state.
		
		If the value is `YES`, QualNet will advance the simulation time in "Pause" state.
		
		If the value is `NO`, QualNet will not advance the simulation time in "Pause" state.

`SOCKET-INTERFACE-ONLY-`
`DEACTIVATE-MAPPED-`
`NODES`
 	
Optional
 	
Scope: Global
	List:
	• `YES`
	• `NO`
	
	Default: `NO`
		Indicates whether QualNet should deactivate only nodes mapped in the entity mapping file during start-up.
		
		If the value is `YES`, QualNet will only deactivate mapped nodes during start-up
		
		If the value is `NO`, QualNet will deactivate all nodes in the scenario during start-up

3.3.1.1 Format of the Entity Mapping File

The entity mapping file specifies static mappings between the external entities and QualNet nodes. Several QualNet nodes can map to the same external entity. Each line in this file has the following format:

```
<External-Entity-ID> <QualNet-Node-List>
```

where

`<External-Entity-ID>`   Identifier (string) for the entity used by the external program.

`<QualNet-Node-List>`   List of `QualNet` nodes that map to the external entity. A node can be referenced by it node ID, hostname, or IP address. Node IDs, hostnames, and IP addresses in this list are separated by spaces.

> If the list contains a node ID, the node with that ID maps to the external entity. If the list contains a hostname, all `QualNet` nodes with that hostname map to the external entity. If the list contains an IP address, the node with that IP address (i.e., the node having that IP address as its interface address) maps to the external entity.

> **Note:** A hostname can be any string (including an integer or an IP address). When matching a node reference in the node list with hostnames, node IDs and IP addresses, a match with a hostname takes precedence over a match with a node ID or IP address.

Note: The entity identifier and node labels can optionally be enclosed within quotes (").

The following is an example of an entity mapping file:

```
# Entity ID        QualNet Node List

1                  10
2                  25 16 radio-101
"Entity 1"         501  192.168.1.1
3                  35 "Platform 1"
Station-10         15 20 35
```

3.3.2 GUI Configuration

To configure the QualNet Socket parameters using the QualNet GUI, do the following:

1. Go to **Scenario Properties Editor > External Interfaces > Socket Interface**.

2. Set the parameters listed in Table 3-33.
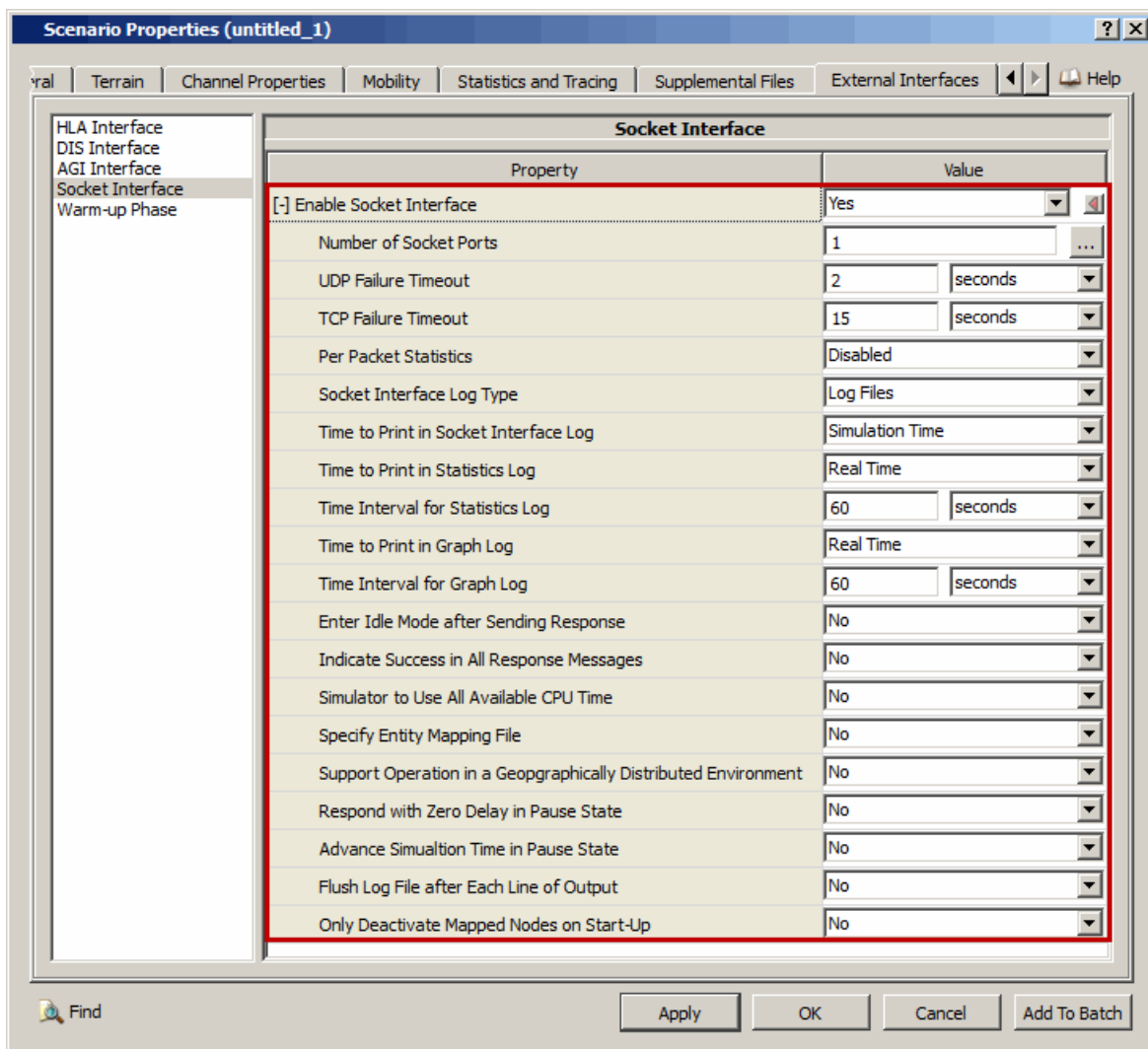
FIGURE 3-3.   Setting Socket Interface Parameters

TABLE 3-33.   Command Line Equivalent of Socket Interface Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Enable Socket Interface	Global	`SOCKET-INTERFACE`
Number of Socket Ports	Global	`SOCKET-INTERFACE-NUM-PORTS`
UDP Failure Timeout	Global	`SOCKET-INTERFACE-UDP-FAILURE-TIMEOUT`
TCP Failure Timeout	Global	`SOCKET-INTERFACE-TCP-FAILURE-TIMEOUT`
Per Packet Statistics	Global	`SOCKET-INTERFACE-PRINT-PER-PACKET-STATS`
Socket Interface Log Type	Global	`SOCKET-INTERFACE-LOG`
Time to Print in Socket Interface Log	Global	`SOCKET-INTERFACE-PRINT-REAL-TIME`
Time to Print in Statistics Log	Global	`SOCKET-INTERFACE-STATS-PRINT-REAL-TIME`
Time Interval for Statistics Log	Global	`SOCKET-INTERFACE-STATS-PRINT-INTERVAL`
Time to Print in Graph Log	Global	`SOCKET-INTERFACE-GRAPH-PRINT-REAL-TIME`
Time Interval for Graph Log	Global	`SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL`
Enter Idle Mode after Sending Response	Global	`SOCKET-INTERFACE-IDLE-WHEN-RESPONSE-SENT`
Indicate Success in All Response Messages	Global	`SOCKET-INTERFACE-ALWAYS-SUCCESS`
Simulator to Use All Available CPU Time	Global	`SOCKET-INTERFACE-CPU-HOG`
Specify Entity Mapping File	Global	N/A
Support Operation in a Geographically Distributed Environment	Global	`SOCKET-INTERFACE-DISTRIBUTED-ENVIRONMENT`
Respond with Zero Delay in Pause State	Global	`SOCKET-INTERFACE-PAUSE-REPLY-ZERO-DELAY`
Advance Simulation Time in Pause State	Global	`SOCKET-INTERFACE-PAUSE-ADVANCE-SIMULATION-TIME`
Flush Log File after Each Line of Output	Global	`SOCKET-INTERFACE-LOG-AUTOMATIC-FLUSH`
Only Deactivate Mapped Nodes on Start-Up	Global	`SOCKET-INTERFACE-ONLY-DEACTIVATE-MAPPED-NODES`

Setting Parameters

- To disable printing of per-packet statistics, set **Per Packet Statistics** to *Disabled*. To print per-packet statistics on the terminal window, set **Per Packet Statistics** to *Terminal WIndow*. To print per-packet statistics to a file, set **Per Packet Statistics** to *File.*

- To disable logging of data, set **CES Log Type** to *Disabled*. To log all output on the terminal window, set **CES Log Type** to *Terminal WIndow*. To log output to files, set **CES Log Type** to *Log Files.*

- To specify an entity mapping file, set **Specify Entity Mapping File** to *Yes*; otherwise, set **Specify Entity Mapping File** to *No*.

3. To configure Socket ports, do the following:

    a. Set **Number of Socket Ports** as shown in Figure 3-3.

    b. Click on the **Open Array Editor** ... button in the **Value** column. This opens the Array Editor.

    c. In the left panel of the Array Editor, select the index of the socket port to be configured. In the right panel, set the parameters listed in Table 3-34.
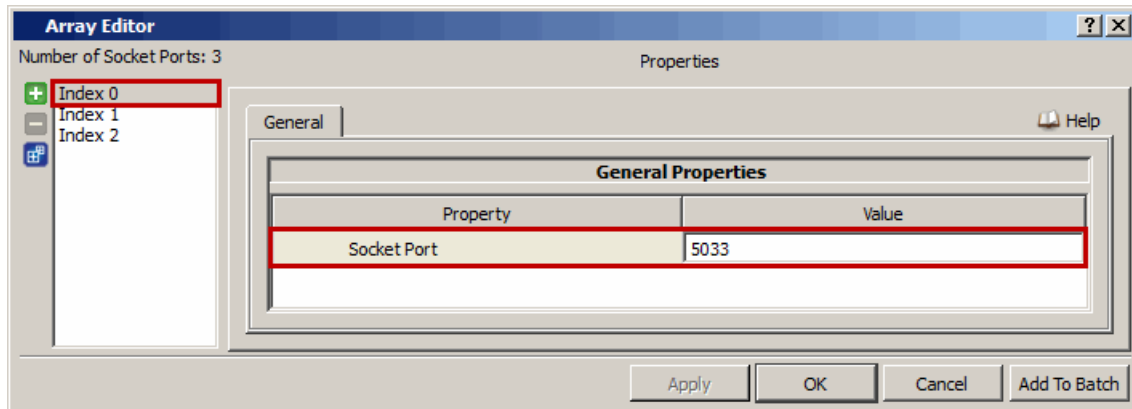
**FIGURE 3-4.   Setting Socket Ports**

TABLE 3-34.   Command Line Equivalent of Socket Port Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Socket Port	Global	`SOCKET-INTERFACE-PORT`

4. If **Specify Entity Mapping File** is set to *Yes*, then set the dependent parameters listed in Table 3-33.
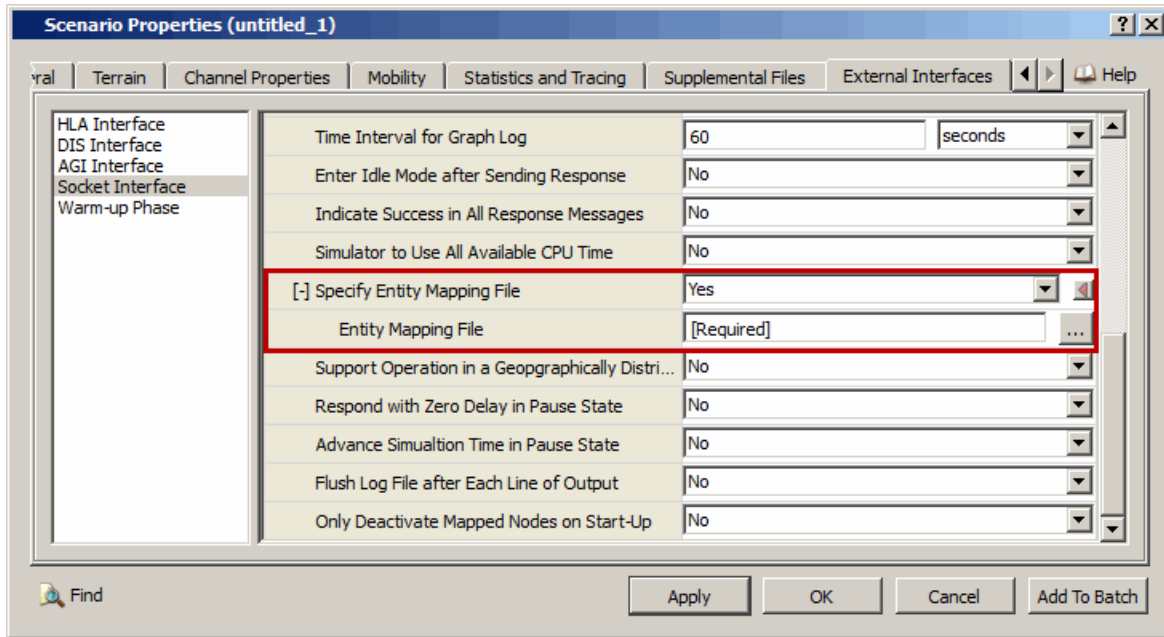
FIGURE 3-5.   Specifying Entity Mapping File

TABLE 3-35.   Command Line Equivalent of Entity Mapping File Parameters

GUI Parameter	Scope of GUI Parameter	Command Line Parameter
Entity Mapping File	Global	`SOCKET-INTERFACE-ENTITY-MAPPING-FILE`

Setting Parameters

- Set **Entity Mapping File** to the name of the entity mapping file. See Section 3.3.1.1 for the format of the entity mapping file.

3.4 Output Files

This section describes the log files that are generated when QualNet Socket is run.

QualNet log files are generated in the directory from which QualNet is run. They are stored for each run in a directory that is named using the following convention:

        CES_SOCKET_<date>_<time>

where

        <date>   Date of the run (year, month, day) in the following format: YYYYMMDD

        <time>   Time of the run (hour, minutes, seconds) in the following format: HHMMSS

Example

> A directory with the name CES_SOCKET_20040802_135822 stores log files for the run that began at 1:58:22 PM on August 2nd, 2004.

The log files generated by QualNet are listed in Table 3-36 and are explained in the following sections.

TABLE 3-36.   QualNet Socket Log Files

Name	Description
driver.log	Messages received by QualNet from the external program.
errors.log	Errors encountered during the run.
graph.log	Network connectivity graph.
responses.log	Messages sent by QualNet to the external program.
stats.log	Performance statistics for QualNet.

3.4.1 File driver.log

The file driver.log records all messages sent by the external program to QualNet. The file contains one line for each message and has the following format:

    <time> [<real-time>] <message_name> [<parameter_list>]

where

`<time>`	Simulation time when the command was sent, enclosed in square brackets.
`<real-time>`	Elapsed real time since the beginning of the simulation. This is only printed if SOCKET-INTERFACE-PRINT-REAL-TIME is set to YES.
`<message_name>`	Message name.
`<parameter_list>`	List of parameters included in the message. If the message does not include any parameters, this list is empty.

If the message has parameters, the parameter list is printed using the following format:

    : <param-name> = <param-value> {,<param-name> = <param-value>}

where

`<param-name>`	Parameter name.
`<param-value>`	Parameter value.

See Section 3.1.2 for a description of interface messages that can be sent from the external program to QualNet and their associated parameters.

For an example of the file driver.log, see Section E.3.3.1.

3.4.2 File errors.log

The file errors.log records all errors that occur during a simulation run. Each error is printed on a line in the following format:

```
<time> ErrorMessage <error_type>, ERROR = <description>
        [originating message = <message>]
```

where

`<time>`	Simulation time when the error occurred, enclosed in square brackets.
`<error_type>`	Type of error.
`<description>`	Description of the error.
`<message>`	Message that caused the error. Some error entries may not include a message with them.

The possible error types that can be printed in the errors.log file are listed in Table 3-26.

For an example of file errors.log, see Section E.3.3.2.

3.4.3 File graph.log

The file graph.log records details of the routing status of scenario's topology. Data are printed to the file graph.log at intervals defined by the parameter `SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL`. After each interval, data are printed for each node in the scenario, followed by a region summary if regions have formed.

Information for each node as well as the region summary can span several lines in the log file. Each line in the file begins with a timestamp which indicates the simulation time when the information was printed.

The information for each entity is printed in the following format:

- The first line indicates the entity identifier (`EntityID`) and the node identifier (`NodeID`) that the entity maps to.
- The second line indicates the node location. If the QualNet scenario uses the Cartesian coordinate system, then the X-, Y-, and Z-coordinates of the node are printed. If the QualNet scenario uses the Lat/Lon/Alt coordinate system, then the latitude, longitude, and altitude of the node are printed.
- The third line indicates the platform type of the node (`Ground` or `Air`). See Section 3.1.2.8.
- Next, information is printed about the node's interfaces. One line is printed for each subnet that the node belongs to and has the following format:
  - The node's interface address and the subnet identifier of the subnet are printed.
  - If regions have formed in the subnet, and the node is a RAP, then the label `[RAP]` is printed on the same line after the subnet identifier, followed by a list of other nodes in the same region. Note that the entity identifiers corresponding to the nodes are printed.
- The next line indicates whether the node is a gateway, and if so, the subnets of which it is a gateway.

A region summary is printed only if regions have formed by that time in simulation. The region summary consists of the following information about each subnet which has formed regions:

- The first line indicates the subnet identifier followed by a list of entity identifiers of all gateways of the subnet.

- Each subsequent line corresponds to a region. All nodes belonging to a region are printed on a single line. The list of RAP nodes is printed first and is enclosed in parentheses.

For an example of file graph.log, see Section E.3.3.3.

3.4.4 File responses.log

The file responses.log records messages sent by QualNet to the external program. The file has the same format as the file driver.log.

See Section 3.1.3 for a description of interface messages sent from the external program to QualNet and their associated parameters.

For an example of the file responses.log, see Section E.3.3.4.

3.4.5 File stats.log

The file stats.log provides snapshots of the Socket Interface by recording performance statistics at fixed time intervals. The time interval is determined by the SOCKET-INTERFACE-STATS-PRINT-INTERVAL parameter in the QualNet configuration file (see Section 3.3).

Each line in the stats.log file prints one or more statistics and starts with a time-stamp (enclosed in square brackets).

For a description of the format and an example of the file stats.log, see Section E.3.3.5.

A Extractor and Synchronizer

For QualNet to provide communication effects to another simulator, it should simulate the same scenario as the one simulated by the other simulator. Extractor and Synchronizer are simple, easy-to-use tools that takes a scenario in other simulators as input and creates an equivalent QualNet scenario that has sufficient detail to simulate the protocol stack and communication capabilities of all communicating entities in the scenario. Extractor (see Section A.6) has a graphical interface whereas Synchronizer (see Section A.7) is run from the command line.

Extractor and Synchronizer use VR-Link to create equivalent QualNet scenarios from other simulator scenarios. With VR-Link's protocol-independent API, they can create scenarios for Distributed Interactive Simulation (DIS) or the High-Level Architecture (HLA).

A.1 System Requirements

System requirements for running and compiling Extractor and Synchronizer are the same as for running and compiling QualNet with VR-Link (see Section 2.1.1).

A.2 Environment Variables for Running Extractor and Synchronizer

The environment variables that need to be set for running Extractor and Synchronizer are the same as those for running QualNet with HLA and DIS (see Section 2.1.3).

A.3 Compiling Extractor and Synchronizer

The Extractor executable file is called Extractor.exe on Windows platforms and Extractor on Linux platforms. The Synchronizer executable file is called synch.exe on Windows platforms and synch on Linux platforms. These files are placed in the directory QUALNET_HOME/bin when the Federation Interfaces Library is installed. These executables are recreated whenever QualNet is recompiled with VR-Link.

If you need to recreate the Extractor or Synchroniser executable file, recompile QualNet with VR-Link (see Section 2.1.5 for details).

A.4 Settings Files

Extractor and Synchronizer use two types of settings files to create a QualNet scenario:

- **System Parameter Files:** These files (which have the extension ".vrlink") contain the general simulation configuration parameters.
- **Router Model Files:** These files (which have the extension ".router-models") contain definitions of pre-configured QualNet models for several types of network devices. Parameters specifying the hardware and software capabilities of the network device are associated with each router model.

These settings files are located in QUALNET_HOME/gui/devices.

> **Note:** Users can modify the settings files to customize the QualNet scenario generated by Extractor or Synchronizer.

A.5 Extraction Rules

The QualNet scenario created by Extractor or Synchronizer has sufficient detail to simulate mobility and communication between entities within the federation. For each entity in the federation that has communication capabilities, Extractor and Synchronizer create a node in the QualNet scenario. Section A.5.1 describes how the properties of nodes in the QualNet scenario are configured, and describes the rules for defining the topology in the QualNet scenario.

A.5.1 Configuring Nodes

An entity and its associated radio in a federation are mapped to a single QualNet node. If an entity has multiple associated radios, then it gets mapped to as many QualNet nodes as the number of associated radios. Extractor and Synchronizer ignore entity objects that do not have radios and radio objects that do not have entities. Each radio has a type, which is defined using the six-field DIS/HLA Radio Entity Type Record. This type is used as an index to determine the network parameters QualNet will use to model the radio.

In a router model, the `VRLINK-RADIO-SYTEM-TYPE` parameter is used to determine which radio types map to that router model. A value of -1 for the Radio Entity Type Record field acts as a wild card and matches any value.

For example, the HIGH-POWER-802.11b router model uses the default radio type for VR-Forces, 7.1.225.2.1.20, to assign PHY 802.11b parameters to that radio type.

```
ROUTER-MODEL                    HIGH-POWER-802.11b
VRLINK-RADIO-SYSTEM-TYPE        7.1.225.2.1.20
PHY-MODEL                       PHY802.11b
PHY-RX-MODEL                    PHY802.11b
PHY802.11-AUTO-RATE-FALLBACK    NO
```

A.5.2 Defining Topology

Extractor and Synchronizer try to create a reasonable network topology and assigns nodes to subnets based on the following rules:

- Entities with different Force IDs are placed in different subnets.
- Entities with incompatible MAC or network protocols are placed in different subnets.
- Aggregate Entities are mapped to QualNet hierarchies.
- Entities in different hierarchies are placed in different subnets.
- Entities that are not explicitly members of an Aggregate Entity are placed in the root hierarchy.

A.6 Using Extractor

To create a QualNet scenario using Extractor, perform the following steps:

1. If you want to create a scenario for HLA, start the RTI.

2. Start the other simulator(s) in the federation, such as VR-Forces, and load the scenario you wish to extract.

3. Go to QUALNET_HOME/bin and launch Extractor. The name of the Extractor executable file is Extractor.exe on Windows platforms and Extractor on Linux platforms.

4. On Extractor's welcome screen, click **Next** to begin setup.
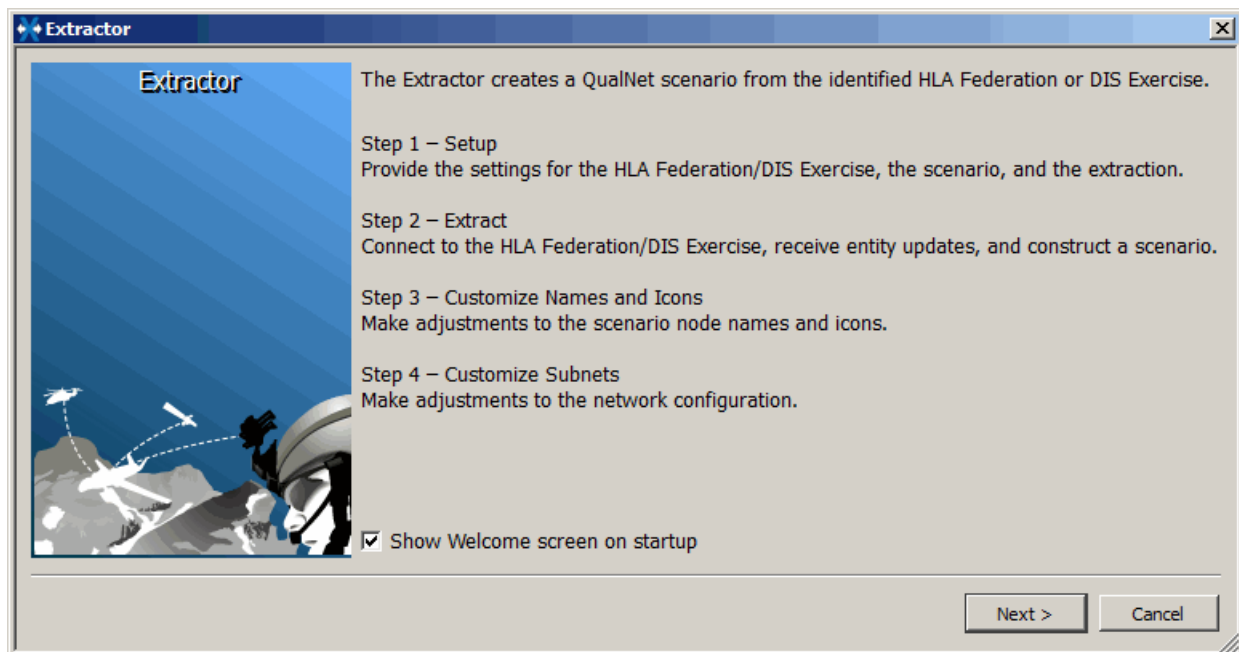
FIGURE A-1.   Extractor Welcome Screen

5. The **Input Setup** tab of the **Setup** screen is displayed which is used for specifying the input parameters for extraction (see Figure A-2).

6. Under **Choose Protocol**, select a protocol from one of the options: **HLA 1.3**, **HLA 1516**, or **DIS**.
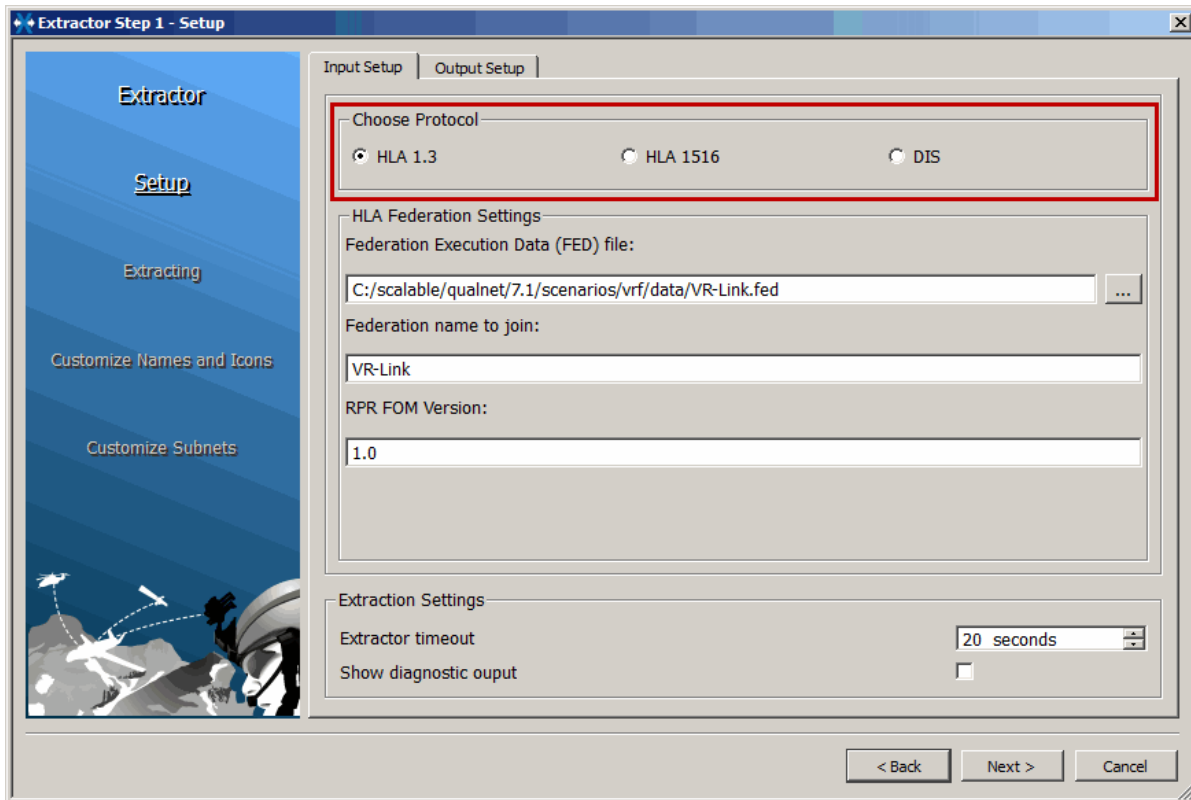
FIGURE A-2.   Input Setup Tab

7. For HLA 1.3 or HLA 1416, under **HLA Federation Settings** (see Figure A-3), set the following parameters:

- **Federation Execution Data (FED) file** (for HLA 1.3) or **Federation Document Data (FDD) file** (for HLA 1516): Specify the name of the of the Federation Execution Data (FED) or Federation Execution Data (FED) file. The FED or FDD file is required as an input to the RTI.

- **Federation name to join**: Enter the name of the HLA federation to join. This must exactly match the federation name used by the other simulators.

- **RPR FOM Version**: Enter the Real-time Platform Reference Federation Object Model (RPR FOM) version. If this field is left blank, RPR FOM version 1.0 is used by default. (Extractor supports only RPR FOM.)
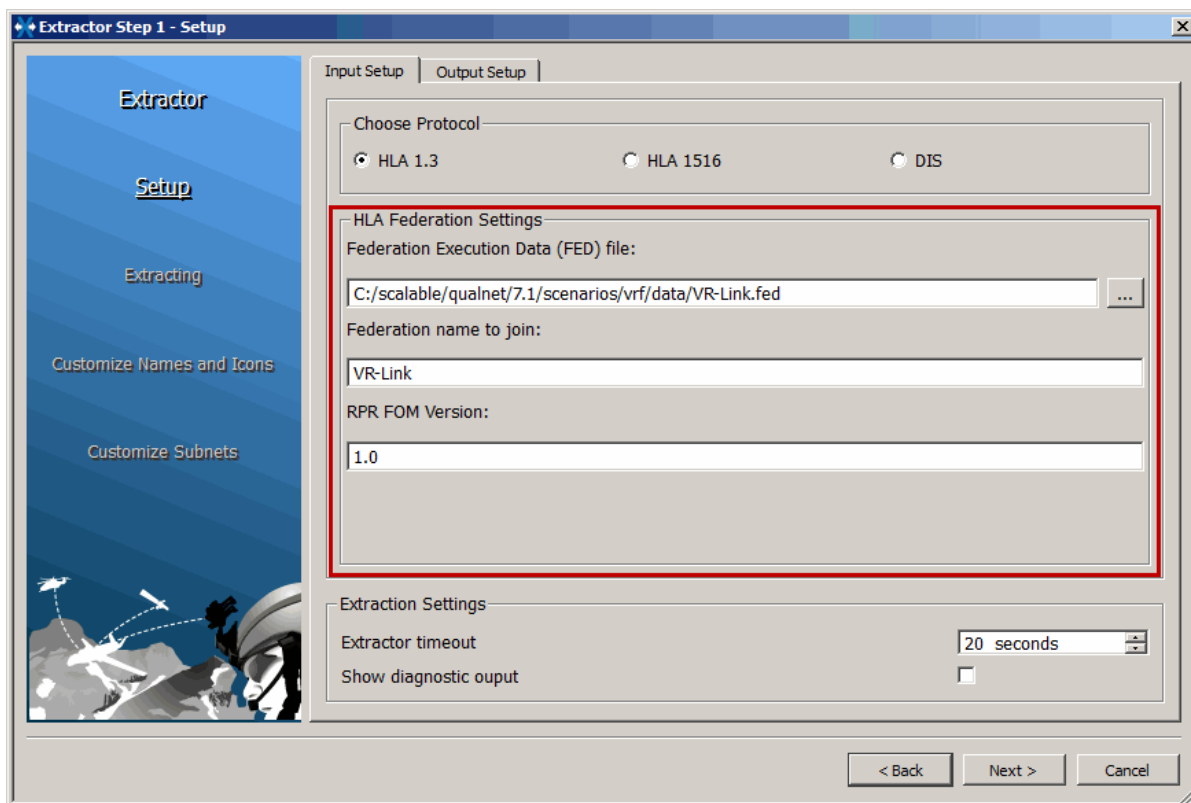
FIGURE A-3.   Input Settings for HLA

8. For DIS, under **DIS Exercise Settings** (see Figure A-4), set the following parameters:

- **Entering Port**: Enter the DIS port number to be used.
- **Destination Address**: Enter the destination IP address.
- **Network Device Address**: Enter the network device's IP address.
- **Set Subnet Mask**: To specify the subnet mask for the DIS socket used for listening and sending, check this box and enter the subnet mask in the field that is displayed.
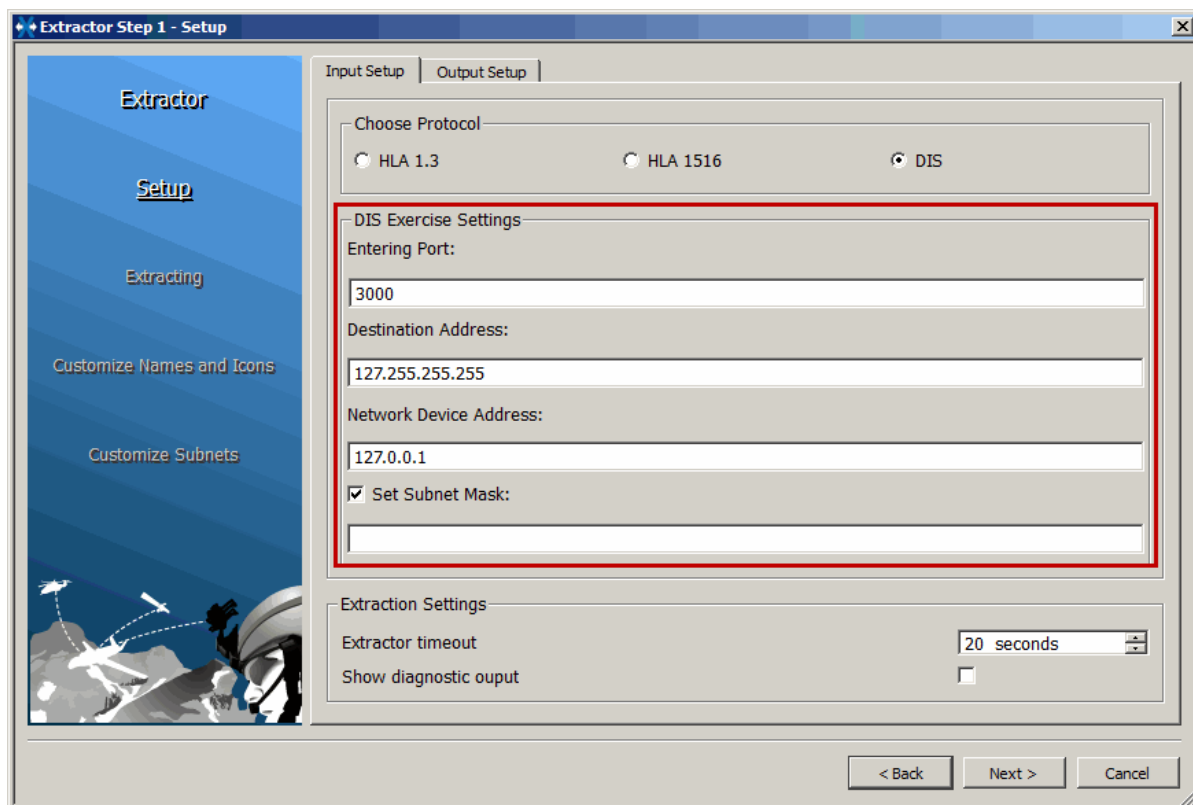
FIGURE A-4. Input Settings for DIS

9. For both HLA and DIS, under **Extraction Settings** (see Figure A-5), set the following parameters:

- **Extractor timeout**: The Extractor timeout controls how long the Extractor will stay connected to the HLA federation or DIS exercise and collect updates. When the timeout expires, Extractor will create the initial QualNet scenario and automatically advance to the customization steps. If the timeout is too short, Extractor may miss updates and the resulting scenario may contain errors.

  To set a timeout of infinity, set the **Extractor timeout** to None by repeatedly clicking the down arrow or enter 0 as the timeout value. (If the timeout is set to infinity, Extractor will not advance to the customization steps until you click on the **Next** button, as explained later.)

- **Show diagnostic output**: Check this box to display status messages during the extraction. These messages may help diagnose problems in the extraction process.
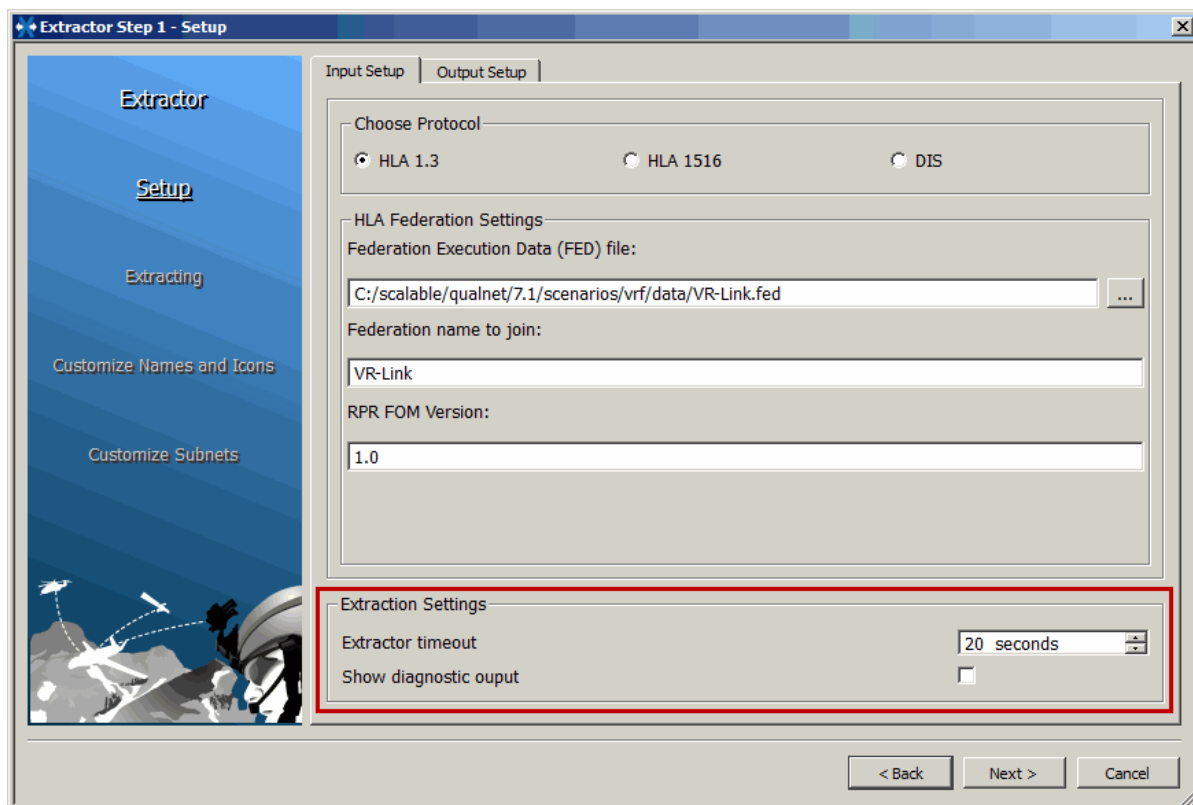
FIGURE A-5.   Extraction Settings

10. In the **Output Setup** tab, under **QualNet Scenario Settings** (see Figure A-5), set the following parameters:

- **Scenario name**: Enter the name of the QualNet scenario to be created.
- **Scenario directory**: Specify the directory where the files for QualNet scenario should be placed.
- **Copy of the FED/FDD file into the scenario directory** (only for HLA 1.3 and HLA 1516): Check this box to place a copy of the FED or FDD file in the QualNet scenario directory.

  It is recommended to use this option since it makes it easier for the QualNet simulator to find and access the FED or FDD file.
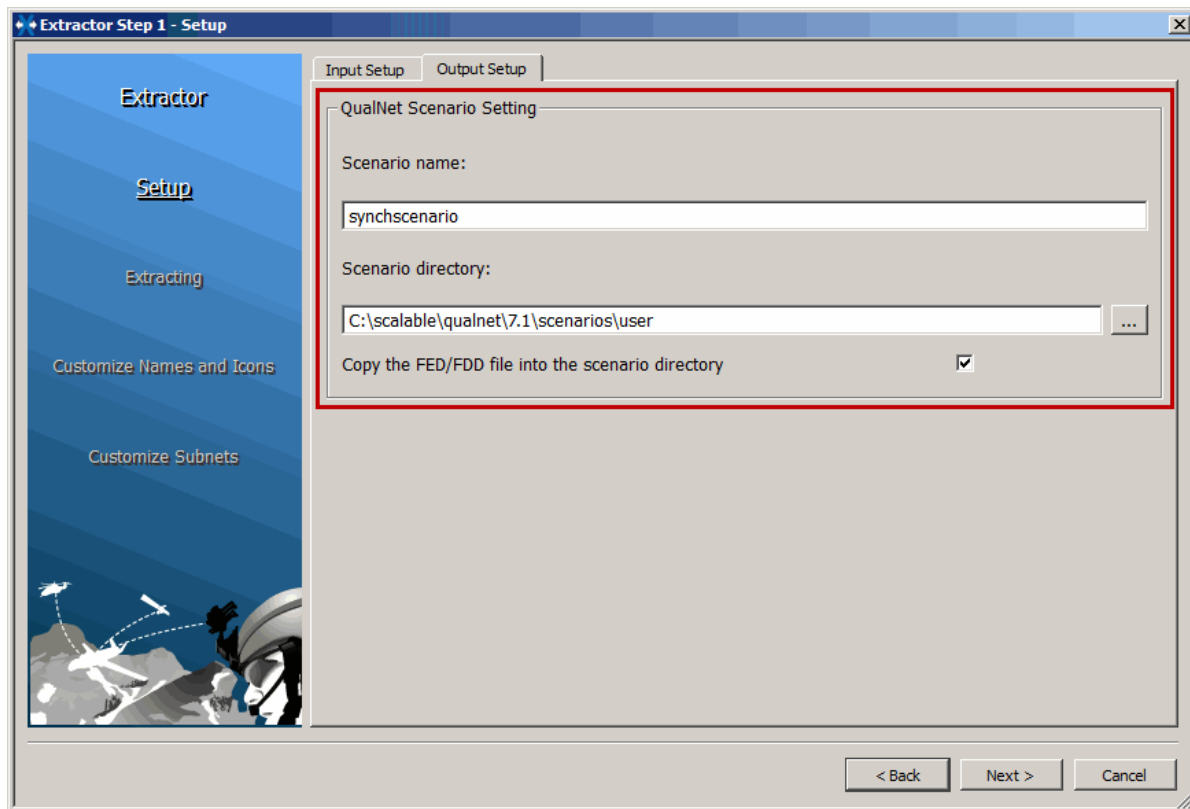
FIGURE A-6.   Output Settings

11. After setting the parameters, click **Next** to start the extraction.

12. If you are using HLA, a RTI connection dialog may open. Select the RTI connection to use and click **Connect**. (Figure A-7 shows the connection dialog for MAK RTI.)
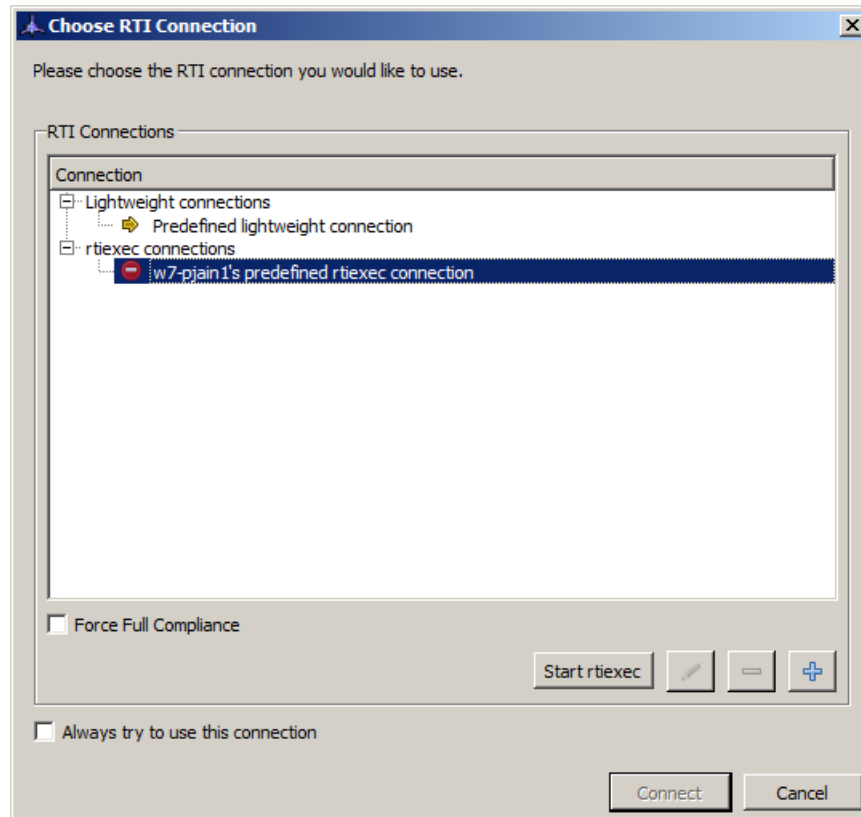
FIGURE A-7.   MAK RTI Connections Dialog

13. After the RTI connection is established, the **Extracting** screen is displayed. Extractor extracts information and displays the status in the **Extracting** screen.
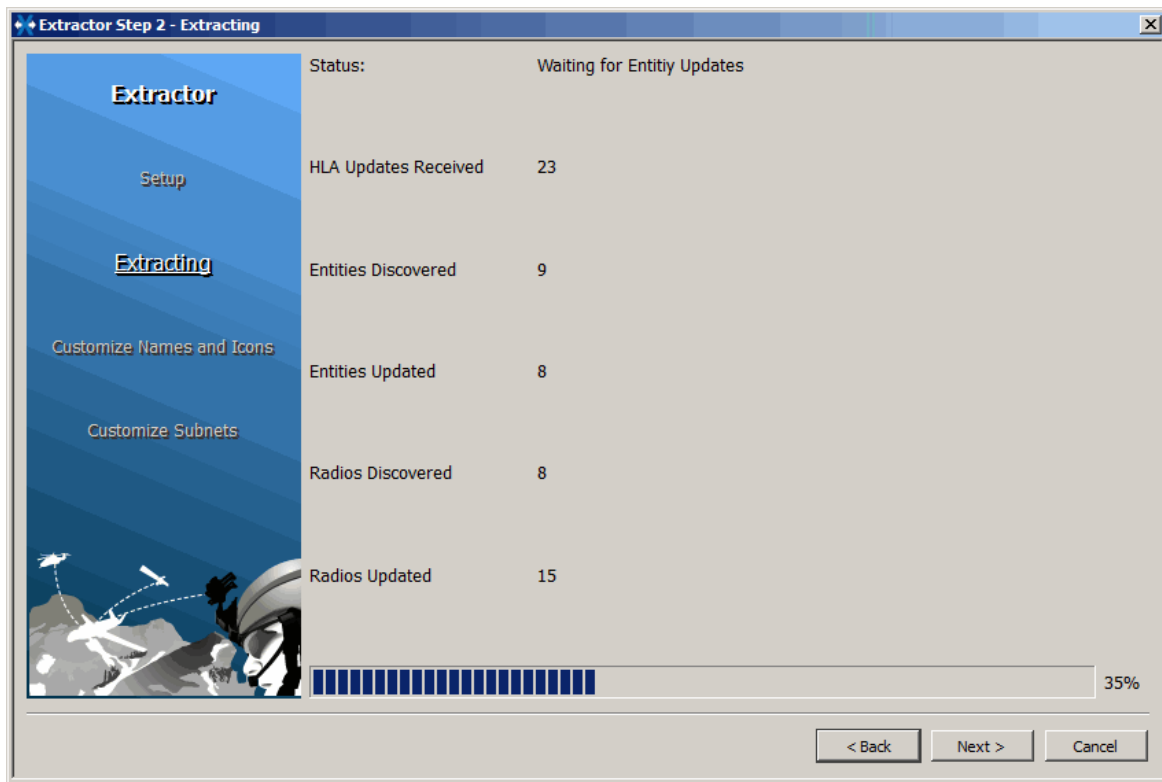
FIGURE A-8.   Extraction Information

If you selected the option to show diagnostic output, status messages are displayed in an output window while the scenario is extracted.
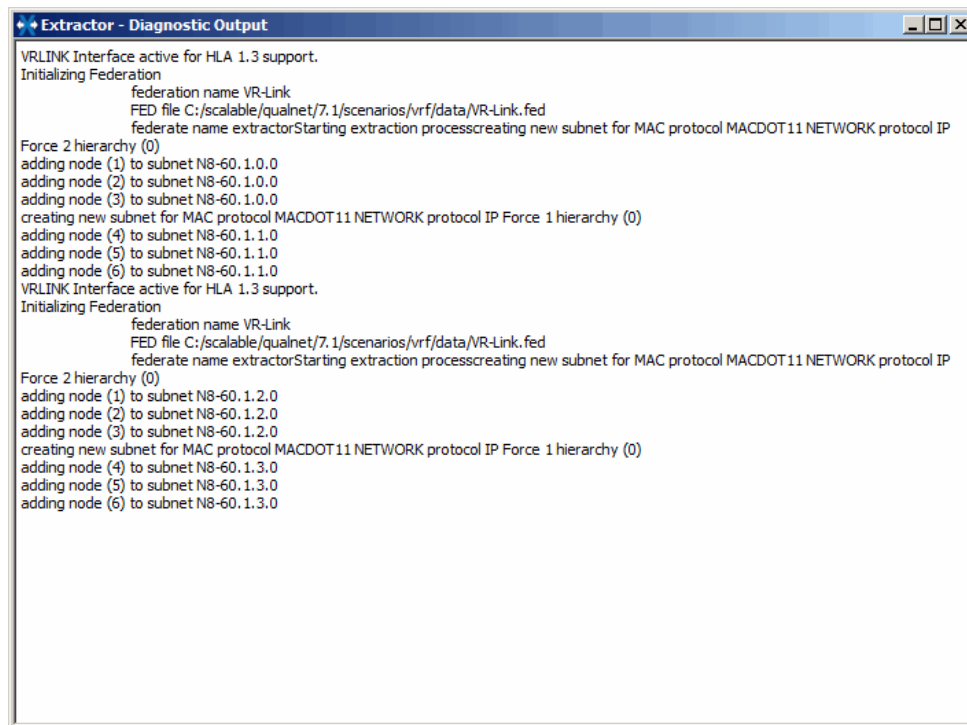
FIGURE A-9.   Diagnostic Output

The progress bar in the **Extracting** screen displays the elapsed time as a percentage of the Extractor timeout. When the timeout expires, Extractor will create the initial QualNet scenario and automatically advance to the customization steps. To advance to the customization steps before the timeout expires, click **Next**.

 Note: Clicking **Next** will terminate the extraction process.

14. When the Extractor timeout has expired or you have terminated the extraction by clicking **Next**, the **Customize Names and Icons** screen is displayed which shows the list of extracted nodes with the assigned names and icons. Extractor assigns names to QualNet nodes that are made available to the federation. From this screen you can override the names and icons assigned to QualNet nodes.
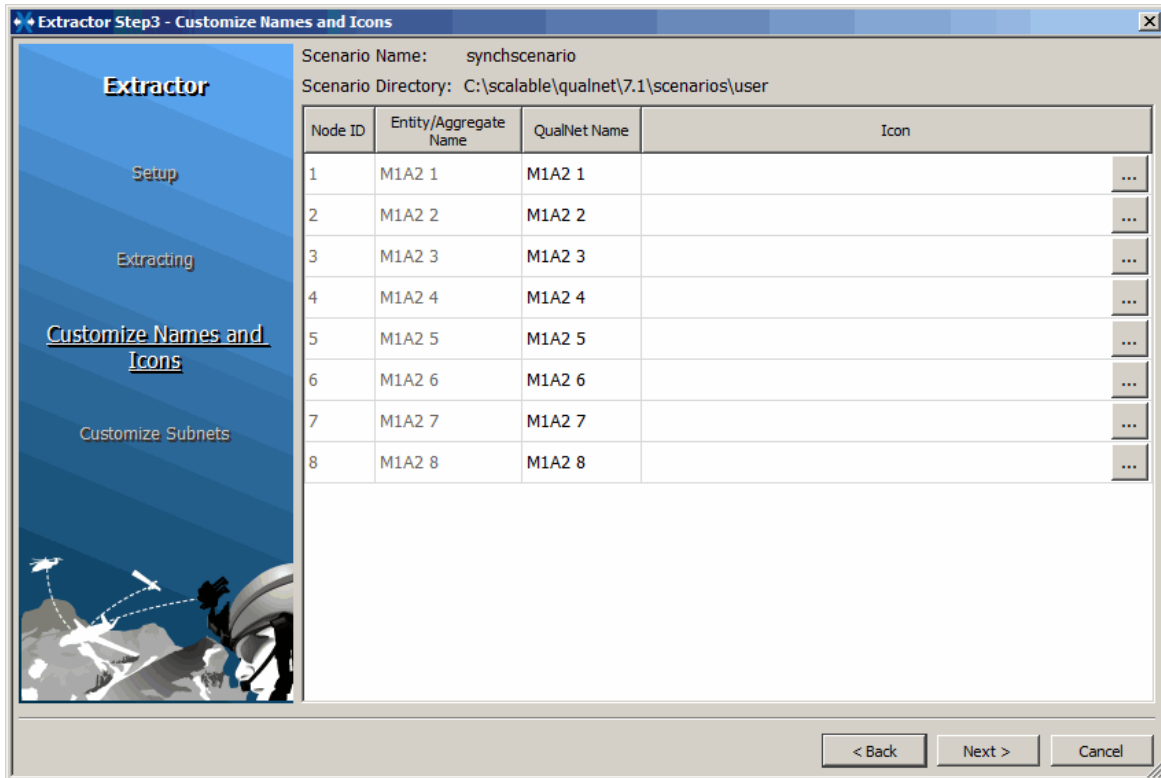
FIGURE A-10.   Customizing Names and Icons

To change the name of a node, type the new name in the **QualNet Name** field. To change a node icon, specify the image file to use in the **Icon** field.

After making the desired changes, click **Next**.

15. The **Customize Subnets** screen is displayed which shows the node names and the subnets to which the nodes are assigned. Extractor uses the rules described in Section A.5.2 to assign nodes to subnets. From this screen you can change the subnet to which a node is assigned.
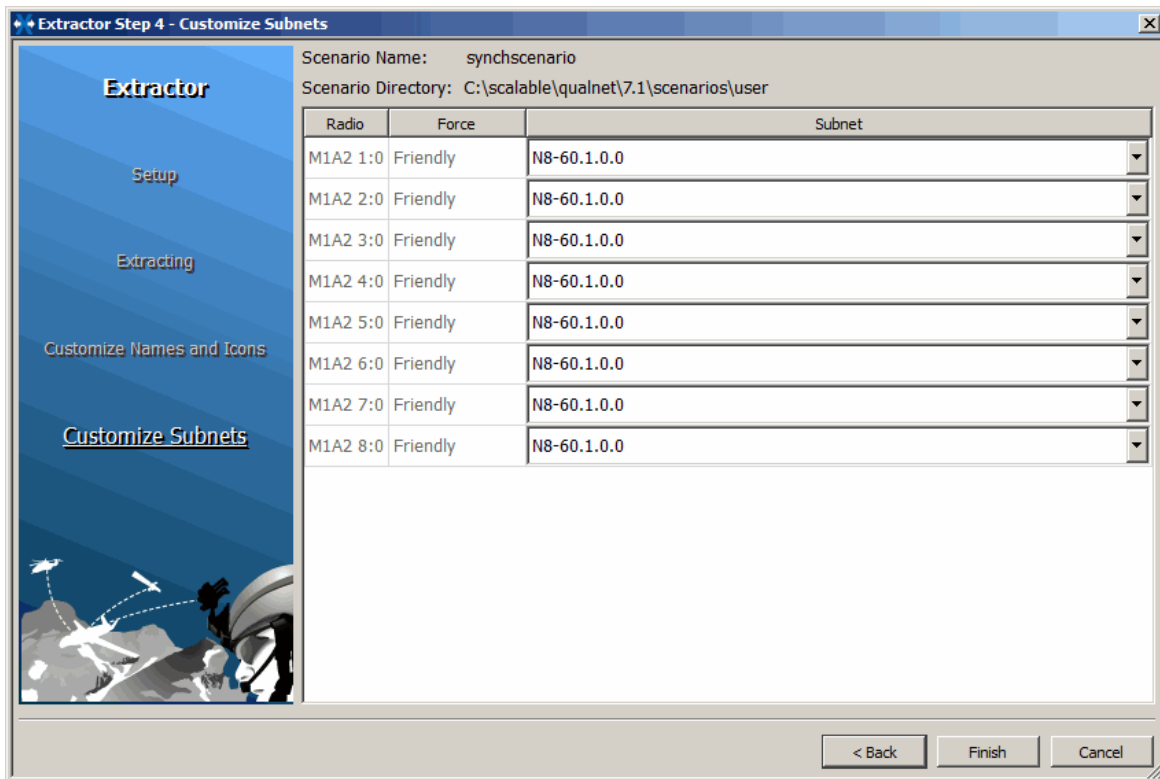
FIGURE A-11.   Customizing Subnets

To change the subnet assignment of a node, select a subnet from the drop-down list.

After making the desired changes, click **Finish**.

Generated Files

The following QualNet scenario files are created in the specified folder (<scenario-name> is the name of the scenario entered in the **Scenario Name** field):

- Scenario Configuration File (<scenario-name>.config)
- Node Placement File (<scenario-name>.nodes)
- Router Models File (<scenario-name>.router-models)
- Entities File (<scenario-name>.vrlink-entities)
- Radios File (<scenario-name>.vrlink-radios)
- Networks File (<scenario-name>.vrlink-networks)

▪▪

A.7 Using Synchronizer

Synchronizer is run from the command line to create a QualNet scenario and can be particularly useful for batch file processing.

> **Note:** If Synchronizer is used to create a QualNet scenario, the name, icon, and subnet assignments of nodes can not be changed directly. The users can edit the generated scenario configuration (.config) file to change these assignments.

A.7.1 Creating HLA Scenarios Using Synchronizer

To create an HLA scenario using Synchronizer, do the following:

1. Start the other simulator(s) in the federation, such as VR-Forces, and load the scenario you wish to extract.

2. Start the RTI, if it is not already running.

3. Open a command window, navigate to QUALNET_HOME/bin, and type the following command:

```
synch  -P <federation-protocol>
       [-F <FED-or-FDD-file>] [-r <FOM-version>] [-f <federation-name>]
       [-N <node ID>] [-s <network-number>] [-t <timeout>]
       [<scenario-name>]
```

> **Note:** All parameters must be entered on one line.

These parameters are explained in Table A-1.

TABLE A-1.   Synchronizer Parameters for HLA

Parameter	Description
`-P <federation-protocol>` *Required*	To use HLA, set `<federation-protocol>` to `HLA13` or `HLA1516`, depending on the HLA standard.
`-F <FED-or-FDD-file>` *Required*	If you are using HLA 1.3, set `<FED-or-FDD-file>` to the name of the FED (.fed) file. If you are using HLA 1516, set `<FED-or-FDD-file>` to the name of the FDD (.xml) file.
`-r <FOM-version>` *Optional*	Set `<FOM-version>` to the RPR FOM version (`1.0` or `2.0017`). If this parameter is not specified, RPR FOM 1.0 is used by default.
`-f <federation-name>` *Optional parameter*	Set `<federation-name>` to the name of the federation to join. If this parameter is not specified, the default federation name is `RPR-FOM`.
`-N <node-ID>` *Optional*	Set `<node-ID>` to the initial node ID. If this parameter is not specified, the default initial node ID is `1`.
`-s <network-number>` *Optional*	Set `<network-number>` to the class A network number for new networks. If this parameter is not specified, the default class A network number for new networks is `60.1.0.0`.

TABLE A-1.   Synchronizer Parameters for HLA (Continued)

Parameter	Description
`-t <timeout>` *Optional*	Set `<timeout>` to the number of seconds since the last entity is discovered before the program exits. If this parameter is not specified, the default timeout period is `5`.
`<scenario-name>` *Optional*	Set `<scenario-name>` to the name of the scenario to be created. If this parameter is not specified, the default scenario name is `synchscenario`.

A.7.2 Creating DIS Scenarios Using Synchronizer

To create a DIS scenario using Synchronizer, do the following:

1. Start the other simulator(s) in the federation, such as VR-Forces, and load the scenario you wish to extract.

2. Open a command window, navigate to QUALNET_HOME/bin, and type the following command:

```
synch  -P <federation-protocol>
       [-A <IP-address>] [-I <device-address>]
       [-O <port-number>] [-M <subnet-mask>]
       [-N <node ID>] [-s <network-number>] [-t <timeout>]
       [<scenario-name>]
```

Note: All parameters must be entered on one line.

These parameters are explained in .

TABLE A-2.   Synchronizer Parameters for DIS

Parameter	Description
`-P <federation-protocol>` *Required*	To use DIS, set `<federation-protocol>` to DIS.
`-A <IP-Address>` *Optional*	Set `<IP-address>` to the destination IP address. If this parameter is not specified, the default destination IP address is `127.255.255.255`.
`-I <device-Address>` *Optional*	Set `<device-address>` to the DIS device address. If this parameter is not specified, the default DIS device address is `127.0.0.1`.
`-O <port-number>` *Optional*	Set `<port-number>` to the DIS port number. If this parameter is not specified, the default DIS port number is `3000`.
`-M <subnet-mask>` *Optional*	Set `<subnet-mask>` to the subnet mask for the DIS socket used for listening and sending.
`-N <node-ID>` *Optional*	Set `<node-ID>` to the initial node ID. If this parameter is not specified, the default initial node ID is `1`.

TABLE A-2.   Synchronizer Parameters for DIS (Continued)

Parameter	Description
`-s <network-number>` *Optional*	Set `<network-number>` to the class A network number for new networks. If this parameter is not specified, the default class A network number for new networks is `60.1.0.0`.
`-t <timeout>` *Optional*	Set `<timeout>` to the number of seconds since the last entity is discovered before the program exits. If this parameter is not specified, the default timeout period is `5`.
`<scenario-name>` *Optional*	Set `<scenario-name>` to the name of the scenario to be created. If this parameter is not specified, the default scenario name is `synchscenario.`

A.7.3 Generated Files

The following QualNet scenario files are created in the QUALNET_HOME/scenarios/user/<scenario-name> directory when Synchronizer is used to create a scenario:

- Scenario Configuration File (<scenario-name>.config)
- Node Placement File (<scenario-name>.nodes)
- Router Models File (<scenario-name>.router-models)
- HLA Entities File (<scenario-name>.hla-entities)
- HLA Radios File (<scenario-name>.hla-radios)
- HLA Networks File (<scenario-name>.hla-networks)

B testfed

testfed (test federate) is a command-line based program included in the QualNet distribution which is primarily used to test the QualNet VR-Link interface without the overhead of running a full constructive simulation tool.

testfed creates entities and radios based on input files, and can also request and receive results for communications effects from QualNet. testfed assigns attributes indicating locations, orientations, EntityType attributes, RadioSystemType attributes, etc., based on the input files. Input files for testfed are the same as the files associated with a QualNet scenario: scenario configuration (.config) file, application configuration (.app) file, node position (.nodes) file, etc.

testfed can also be used just to receive outgoing QualNet HLA or DIS traffic. To use testfed as purely a listener, use a testfed command file (see Section B.5) that has only one command: `Quit` command with the a time value greater than the scenario simulation time. testfed can be used as a listener in a federation with QualNet and VR Forces, for example.

B.1 System Requirements

System requirements for running and compiling testfed are the same as for running and compiling QualNet with VR-Link (see Section 2.1.1).

B.2 Environment Variables for Running testfed with HLA and DIS

The environment variables that need to be set for running testfed with HLA and DIS are the same as those for running QualNet with HLA and DIS (see Section 2.1.3).

∙∙

B.3 Compiling testfed

The tstfed executable file (called testfed.exe for Windows platforms and testfed for Linux platforms) is placed in the directory QUALNET_HOME/bin when the Federation Interfaces Library is installed. This executable is recreated whenever QualNet is recompiled with VR-Link.

If you need to recreate the testfed executable file, recompile QualNet with VR-Link (see Section 2.1.5 for details).

∙∙

B.4 Creating Scenarios for testfed

testfed uses the same scenario files as QualNet. Refer to *QualNet User's Guide* for details of configuring scenarios in QualNet. In addition, the parameters described in Section 2.2.2 also need to be configured.

∙∙

B.5 Creating testfed Command File

testfed interacts with the QualNet simulation by means of commands. These commands are entered (one per line) in a testfed command file. The name of the testfed command file should be the same as the QualNet scenario configuration (.config) file used in the exercise but should have the extension ".testfed", for example, if the QualNet scenario configuration file is called multicast-scenario.config, then the testfed command file should be named multicast-scenario.testfed. The testfed command file should be placed in the same directory as the QualNet scenario configuration file.

This commands that can be entered in testfed command (.testfed) file to interact with the QualNet simulation are described below.

- **Register Objects Command**

    The following command registers all objects:

        Register <time>

    where

        <time>   Time (measured from the instant when testfed is started) when the
                            command is sent to QualNet, specified in QualNet time format.

    Example: Register 0S

- **Move Entity Command**

  The following command moves the entity that hosts the radio:

  ```
  Move <time> <node-ID> <lat> <lon> <alt>
  ```

  where

`<time>`	Time (measured from the instant when testfed is started) when the command is sent to QualNet, specified in QualNet time format.
`<node-ID>`	Node ID.
`<lat>`	Latitude (in degrees) of the new position.
`<lon>`	Longitude (in degrees) of the new position.
`<alt>`	Altitude (in meters) of the new position.

  Example: `Move 3S 5 0.01 0.01 20`

- **Change Entity Orientation Command**

  The following command changes the orientation of all radios hosted by the entity:

  ```
  Orientation <time> <node-ID> <psi> <theta> <phi>
  ```

  where

`<time>`	Time (measured from the instant when testfed is started) when the command is sent to QualNet, specified in QualNet time format.
`<node-ID>`	Node ID.
`<psi>,<theta>, <phi>`	Euler angles (in degrees) of the new orientation in the world coordinate system.

  Example: `Orientation 3.5S 5 1.1 1.1 1.0`

- **Change Entity Velocity Command**

  The following command changes the velocity of the entity that hosts the radio:

  ```
  Velocity <time> <node-ID> <x-velocity> <y-velocity> <z-velocity>
  ```

  where

`<time>`	Time (measured from the instant when testfed is started) when the command is sent to QualNet, specified in QualNet time format.
`<node-ID>`	Node ID.
`<x-velocity>`	New velocity (in meters/sec) along the X-axis of the world coordinate system.
`<y-velocity>`	New velocity (in meters/sec) along the Y-axis of the world coordinate system.
`<z-velocity>`	New velocity (in meters/sec) along the Z-axis of the world coordinate system.

  Example: `Velocity 2.5S 3 1.0 1.5 2.0`

- **Change Entity Damage State Command**

  The following command changes the damage state of the entity:

  ```
  Damage <time> <node-ID> <damage-state>
  ```

  where

`<time>`	Time (measured from the instant when testfed is started) when the command is sent to QualNet, specified in QualNet time format.
`<node-ID>`	Node ID.
`<damage-state>`	Damage state (`0`, `1`, `2`, or `3`).

`0`	Not damaged
`1`	Slightly damaged: A 25% chance of up to 75% reduction in transmission power
`2`	Severely damaged: A 75% chance of up to 75% reduction in transmission power
`3`	Destroyed

  Example: `Damage 50S 3 2`

- **Send Communication Effects Request Command**

    The following command sends a communication effects request to QualNet:

    ```
    SendCommRequest <time> <source-ID> [[V]<message-size>]
                    [<timeout-delay>] [<destination-ID>]
    ```

where

`<time>`	Time (measured from the instant when testfed is started) when the command is sent to QualNet, specified in QualNet time format.
`<source-ID>`	Node ID of the source node.
`<message-size>`	Message size (in bytes).
	This parameter is optional. If it is not specified, the default message size is 100 bytes.
	If the message size is preceded by the flag v, it indicates voice traffic.
`<timeout-delay>`	Timeout delay (in seconds).
	The timeout delay is the maximum time QualNet will use to determine if the message was successfully delivered.
	If the communication is successful, QualNet sends a Process Message interaction indicating that the message was delivered to the destination. If there are multiple recipients, each successful delivery will generate a Process Message interaction.
	Whether or not the communication is successful, QualNet sends a Timeout interaction summarizing the delivery status. This occurs when the processing of a message is complete: either after the timeout value or after all potential recipients have processed the signal, whichever occurs first
	This parameter is optional. If it is not specified, the default timeout delay is 10 seconds.
`<destination-ID>`	Node ID of the destination node.
	This parameter is optional. If it is not specified, the packet is sent to the default address for the network, as defined in the Networks file (see Section 2.2.2.1.3).

Example: `SendCommRequest 3.5S 1 24 10 2`

- **Change Radio Transmitter Status Command**

    The following command changes the transmitter status of the radio:

        TxState <time> <node-ID> <transmitter-state>

    where

    `<time>`   Time (measured from the instant when testfed is started) when the
                               command is sent to QualNet, specified in QualNet time format.

    `<node-ID>`   Node ID.

    `<transmitter-state>`   Transmitter state (`0`, `1`, or `2`).

                                          `0`   Off
                                          `1`   On but not transmitting
                                          `2`   On and transmitting

    Example: `TxState 3.5S 4 2`

- **Quit Command**

    The following command ends the testfed session:

        Quit <time>

    where

    `<time>`   Time (measured from the instant when testfed is started) when the
                               command is sent to QualNet, specified in QualNet time format.

    Example: `Quit 30S`

· ·
B.6 Running testfed

This section describes how to run testfed with QualNet using HLA (see Section B.6.1) and using DIS (see
Section B.6.2).

B.6.1 Running testfed Using HLA

To run testfed with QualNet using HLA, perform the following steps:

1. Create the testfed scenario (see Section B.4) and the testfed command file (see Section B.5).

2. Start the RTI, if needed. Some light-weight RTIs do not need to be started explicitly. Refer to the RTI
documentation for details.

3. Open a command window and change the directory to the location where the scenario is located.

4. Type the following command to run testfed (all parameters should be entered on one line):

```
testfed -P <federation-protocol> -F <FED-or-FDD-file>
        [-r <FOM-version>] [-f <federation-name] [-t <vrlink-tick>]
        [-d] <input-file>
```

These parameters are described in Table B-1.

> **Note:** To list the testfed parameters, type **testfed ?** or **testfed h**.

TABLE B-1.   testfed Parameters for HLA

Parameter	Description
-P <federation-protocol> *Required*	To use HLA, set **<federation-protocol>** to HLA13 or HLA1516, depending on the HLA standard.
-F <FED-or-FDD-file> *Required*	If you are using HLA 1.3, set **<FED-or-FDD-file>** to the name of the FED (.fed) file. If you are using HLA 1516, set **<FED-or-FDD-file>** to the name of the FDD (.xml) file.
-r <FOM-version> *Optional*	Set **<FOM-version>** to the RPR FOM version (1.0 or 2.0017). If this parameter is not specified, RPR FOM 1.0 is used by default.
-f <federation-name> *Optional*	Set **<federation-name>** to the name of the federation to join. If this parameter is not specified, the default federation name is RPR-FOM.
-t <vrlink-tick> *Optional*	Set **<vrlink-tick>** to the interval between successive publication of events by the MAK VR-Link engine to the federation. The interval should be specified in QualNet time format, e.g., 1MS, 2.5S. If this parameter is not specified, the default interval is 50 milliseconds.
-d *Optional*	Include this flag to turn on debugging mode.
<input-file> *Required*	Name of the scenario configuration file (without the extension .config).

5. Start the simulation in QualNet either from the command line or using the GUI. Use the same scenario files as for testfed. Refer to *QualNet User's Guide* for details of running simulations.

> **Note:** If the QualNet simulation is run from the command line, then it must be run from a command window different from the one in which testfed is run.

B.6.2 Running testfed Using DIS

To run testfed with QualNet using DIS perform the following steps:

1. Create the testfed scenario (see Section B.4) and the testfed command file (see Section B.5).

2. Open a command window and change the directory to the location where the scenario is located.

3. Type the following command to run testfed (all parameters should be entered on one line):

```
testfed -P <federation-protocol> [-t <vrlink-tick>] [-O <port-number>]
        [-A <IP-address>] [-I <network-address>] [-M <subnet-mask>]
        [-d] <input-file>
```

These parameters are described in Table B-2.

> **Note:** To list the testfed parameters, type **testfed?** or **testfed h**.

TABLE B-2.   testfed Parameters for DIS

Parameter	Description
-P <federation-protocol> *Required*	To use DIS, set **<federation-protocol>** to `DIS`.
-t <vrlink-tick> *Optional*	Set **<vrlink-tick>** to the interval between successive publication of events by the MAK VR-Link engine to the federation. The interval should be specified in QualNet time format, e.g., `1MS`, `2.5S`. If this parameter is not specified, the default interval is 50milliseconds.
-O <port-number> *Optional*	Set **<port-number>** to the port number for the DIS socket. If this parameter is not specified, the default port number is 3000.
-A <IP-address> *Optional*	Set **<IP-address>** to the IP address used by testfed to send and receive DIS PDUs. If this parameter is not specified, the default IP address is 127.255.255.255.
-I <network-address> *Optional*	Set **<network-address>** to network interface address that the socket is created on. If this parameter is not specified, the default network interface address is 127.0.0.1.
-M <subnet-mask> *Optional*	Set **<subnet-mask>** to the subnet mask for the DIS socket used for listening and sending.
-d *Optional*	Include this flag to turn on debugging mode.
<input-file> *Required*	Name of the scenario configuration file (without the extension `.config`).

4. Start the simulation in QualNet either from the command line or using the GUI. Use the same scenario files as for testfed. Refer to *QualNet User's Guide* for details of running simulations.

> **Note:** If the QualNet simulation is run from the command line, then it must be run from a command window different from the one in which testfed is run.

C Upgrading Scenarios

This appendix describes how to modify scenarios using HLA and DIS created for QualNet 6.1 (or earlier) to run with the current version of QualNet.

•••

C.1 Upgrading HLA Scenarios

To upgrade a QualNet 6.1 (or earlier) scenario using HLA, replace the following line in the scenario configuration (.config) file:

```
HLA YES
```

with the following two lines:

```
VRLINK            YES
VRLINK-PROTOCOL   <HLA-specification>
```

where

    `<HLA-specification>`   This is either `HLA13` (for the HLA 1.3 standard) or `HLA1516` (for the HLA 1516 standard).

In addition, replace all occurrences of the old parameters listed Table C-1 with the corresponding new parameter.

TABLE C-1.   Modified HLA Parameters

Old Parameter Name	New Parameter Name
`HLA-DEBUG`	`VRLINK-DEBUG`
`HLA-DEBUG-2`	`VRLINK-DEBUG-2`
`HLA-FEDERATION-NAME`	`VRLINK-FEDERATION-NAME`
`HLA-FED-FILE-PATH`	`VRLINK-FED-FILE-PATH`
`HLA-FEDERATE-NAME`	`VRLINK-DEBUG-2`
`HLA-XYZ-EPSILON`	`VRLINK-XYZ-EPSILON`

TABLE C-1.   Modified HLA Parameters (Continued)

Old Parameter Name	New Parameter Name
HLA-TICK-INTERVAL	VRLINK-TICK-INTERVAL
HLA-MOBILITY-INTERVAL	VRLINK-MOBILITY-INTERVAL
HLA-ENTITIES-FILE-PATH	VRLINK-ENTITIES-FILE-PATH
HLA-RADIOS-FILE-PATH	VRLINK-RADIOS-FILE-PATH
HLA-NETWORKS-FILE-PATH	VRLINK-NETWORKS-FILE-PATH
HLA-DYNAMIC-STATISTICS	VRLINK-HLA-DYNAMIC-STATISTICS
HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE	VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE
HLA-DYNAMIC-STATISTICS-SEND-NODEID-DESCRIPTIONS	VRLINK-HLA-DYNAMIC-STATISTICS-SEND-NODEID-DESCRIPTIONS
HLA-DYNAMIC-STATISTICS-SEND-METRIC-DEFINITIONS	VRLINK-HLA-DYNAMIC-STATISTICS- SEND-METRIC-DEFINITIONS
HLA-PRINT-RTSS	VRLINK-DEBUG-PRINT-RTSS
HLA-RPR-FOM-VERSION	VRLINK-RPR-FOM-VERSION (see note)

Note: RPR FOM 0.5 is no longer supported. VRLINK-RPR-FOM-VERSION must be set to 1.0 or 2.0017 and VRLINK-FED-FILE-PATH must be set to the appropriate file (see Table 2-1).

∙∙

C.2 Upgrading DIS Scenarios

To upgrade a QualNet 6.1 (or earlier) scenario using DIS, replace the following line in the scenario configuration (.config) file:

```
DIS YES
```

with the following two lines:

```
VRLINK          YES
VRLINK-PROTOCOL  DIS
```

In addition, replace all occurrences of the old parameters listed Table C-2 with the corresponding new parameter.

TABLE C-2.   Modified DIS Parameters

Old Parameter Name	New Parameter Name
DIS-DEBUG-PRINT-COMMS	VRLINK-DEBUG-PRINT-COMMS
DIS-DEBUG-PRINT-COMMS-2	VRLINK-DEBUG-PRINT-COMMS-2
DIS-DEBUG-PRINT-MAPPING	VRLINK- DEBUG-PRINT-MAPPING
DIS-DEBUG-PRINT-DAMAGE	VRLINK- DEBUG-PRINT-DAMAGE

TABLE C-2.   Modified DIS Parameters (Continued)

Old Parameter Name	New Parameter Name
DIS-DEBUG-PRINT-TX-STATE	VRLINK-DEBUG-PRINT-TX-STATE
DIS-DEBUG-PRINT-MOBILITY	VRLINK-DEBUG-PRINT-MOBILITY
DIS-DEBUG-PRINT-TRANSMITTER-PDU	VRLINK-DEBUG-PRINT-TRANSMITTER-PDU
DIS-DEBUG-PRINT-PDUS	VRLINK-DEBUG-PRINT-PDUS
DIS-IP-ADDRESS	VRLINK-DIS-IP-ADDRESS (see note)
DIS-PORT	VRLINK-DIS-PORT
DIS-RECEIVE-DELAY	VRLINK-RECEIVE-DELAY
DIS-MAX-RECEIVE-DURATION	VRLINK-MAX-RECEIVE-DURATION
DIS-XYZ-EPSILON	VRLINK-XYZ-EPSILON
DIS-MOBILITY-INTERVAL	VRLINK- MOBILITY-INTERVAL
DIS-ENTITIES-FILE-PATH	VRLINK-ENTITIES-FILE-PATH
DIS-RADIOS-FILE-PATH	VRLINK-RADIOS-FILE-PATH
DIS-NETWORKS-FILE-PATH	VRLINK-NETWORKS-FILE-PATH

Note: If DIS-IP-ADDRESS is set to a value other than 255.255.255.255, then that address may not be reachable with the default network interface of 127.0.0.1. You must set VRLINK-DIS-NETWORK-INTERFACE to a real network interface address (use the **ipconfig** or **ifconfig** command to determine the address).

D Inter-Simulation Communication Protocols

This appendix describes the details of the protocols used for inter-simulation communication. It is intended for programmers who want to interface their simulator with QualNet using HLA or DIS.

D.1 Real-time Platform Reference Federation Object Model (RPR FOM) 1.0

RPR FOM is popular as it is based on the Distributed Interactive Simulation (DIS) protocol, which preceded HLA as a popular standard for simulation interoperability. RPR FOM and DIS are typically used in force-on-force simulations (dismounted infantry, ground, and airborne platforms with blue, red, and neutral forces).

The version of RPR FOM used is RPR FOM 1.0. (Support for RPR FOM 0.5 has been deprecated.)

Section D.2 also contains details on how some of the RPR-FOM elements are used.

This section describes the necessary set of classes for the QualNet HLA protocol. It is possible to use other FOMs that support these classes. The following sections describe objects and interactions used by QualNet.

Attribute descriptions are from the OMT file for RPR FOM 1.0, available at http://www.sisostds.org.

D.1.1 Objects

QualNet subscribes to the Physical Entity objects shown in Table D-1.

TABLE D-1. Physical Entity Objects

Object Classes and Attributes	Description
BaseEntity.PhysicalEntity	A base class of all discrete platform scenario domain participants.
BaseEntity attributes	
EntityIdentifier	The unique identifier for the entity instance.

TABLE D-1.   Physical Entity Objects (Continued)

Object Classes and Attributes	Description
EntityType	The category of the entity.
Orientation	The angles of rotation around the coordinate axes between the entity's attitude and the reference coordinate system axes (calculated as the Tait-Bryan Euler angles specifying the successive rotations needed to transform from the world coordinate system to the entity coordinate system).
VelocityVector	The rate at which an entity's position is changing over time.
WorldLocation	Location of the entity.
BaseEntity.PhysicalEntity attributes	
DamageState	The state of damage of the entity.
ForceIdentifier	The identification of the force that the entity belongs to.
Marking	A unique marking or combination of characters used to distinguish the entity from other entities. (In practice, each RPR FOM federate may handle duplicate Marking values differently - using the first unique value and ignoring subsequent ones, exiting the program, unexpected program behavior, etc.)

QualNet subscribes to the Radio Transmitter Objects shown in Table D-2.

TABLE D-2.   RadioTransmitter Objects

Object Classes and Attributes	Description
EmbeddedSystem.RadioTransmitter	A device that sends out information encoded in electromagnetic waves in the radio frequency range.
EmbeddedSystem attributes	
EntityIdentifier	The Entity Identifier of the object which this embedded system is a part of.
RelativePosition	The position of the embedded system, relative to the host object's position.
EmbeddedSystem.RadioTransmitter attributes	
Frequency	The radio frequency of transmitted radio signals.
RadioIndex	A number that uniquely identifies this radio transmitter from other transmitters on the host entity.
RadioSystemType	The type of radio transmitter.
TransmitterOperationalStatus	The state of the radio transmitter.

D.1.2 Interactions

QualNet subscribes to the ApplicationSpecificRadioSignal interactions shown in Table D-3.

TABLE D-3.   ApplicationSpecificRadioSignal Interactions

Interaction Classes and Parameters	Description
RadioSignal.ApplicationSpecificRadioSignal	A form of radio signal, which uses an application specific encoding scheme.
RadioSignal.ApplicationSpecificRadioSignal parameters	

TABLE D-3.   ApplicationSpecificRadioSignal Interactions (Continued)

Interaction Classes and Parameters	Description
HostRadioIndex	The ID of the radio transmitting this signal. (This is actually the object name, in string form, of the hosting RadioTransmitter object - not the same as the RadioIndex parameter.)
DataRate	The rate at which the data is being transmitted.
SignalDataLength	The length of the signal data.
SignalData	The signal data.
TacticalDataLinkType	The type of tactical data link used to transmitted this signal (if any).
TDLMessageCount	The number of tactical data link messages contained in this signal.
UserProtocolID	The ID of the user protocol in use.

QualNet subscribes and publishes the data interactions shown in Table D-4.

TABLE D-4.   Data Interactions

Interaction Classes and Parameters	Description
Data	A Simulation Management (SIMAN) interaction designed to acknowledge either a) a DataQuery interaction (in which case the Data interaction contains the results of the query) or b) a SetData interaction (in which case the Data interaction contains the data that the federate was able to set).
Data.OriginatingEntity	The DIS entity ID of the entity or application sending the interaction.
Data.ReceivingEntity	The DIS entity ID of the entity or application which is the intended recipient of the interaction.
Data.RequestIdentifier	This field matches this response with the specific SetData or DataQuery interaction sent by the simulation manager.
Data.FixedDatums	The set of data items (types and values), of fixed length, that the recipient can return for this interaction.
Data.VariableDatumSet	The set of data items (types and values) associated with the interaction.

▪▪▪

D.2 HLA Protocol RPR-FOM 1.0 Extensions Interface Communications Document (ICD)

The "client" is a client federate which sends HLA interactions to QualNet to request modeling of communication effects. QualNet sends HLA interactions to the client to report on results of communications.

uint16 refers to a 16-bit unsigned integer; float32 refers to a 32-bit point floating point number; and so on, using the standard representations in C and C++. All fields are in network byte order (most significant byte occurring first).

HLA RPR FOM 1.0 is assumed in this document, although the interface also supports version 2.0017.

The QualNet HLA protocol does not define any new object or interaction classes to be added to RPR FOM 1.0, per se, but rather uses existing interaction classes, except containing QualNet specific data. To determine that an interaction is specific to itself, QualNet uses the UserProtocolID parameter for ApplicationSpecificRadioSignal interactions, and the DatumID field for Data interactions.

It is assumed that the client federate will create the necessary object instances that correspond to the simulation entities sending and receiving communications.

D.2.1 ApplicationSpecificRadioSignal Interaction

The client sends the ApplicationSpecificRadioSignal interaction to request communication effects modeling from QualNet. The interaction parameters should have the values shown in Table D-5.

TABLE D-5.   ApplicationSpecificRadioSignal Interaction

Parameter Name	Value
DataRate	(as defined in RPR FOM)
HostRadioIndex	(as defined in RPR FOM)
SignalDataLength	(as defined in RPR FOM)
SignalData	Message string (the format is described below)
TacticalDataLinkType	Unused
TDLMessageCount	Unused
UserProtocolID	10000

Note: QualNet uses the UserProtocolID parameter to recognize that the interaction is specific to QualNet.

The **HostRadioIndex** parameter can be used to identify the DIS entity ID and RadioIndex which is the source of a message.

This document uses the "Entity ID" term to be synonymous with the RPR-FOM 1.0 EntityIdentifier attribute: three unsigned 16-bit integers consisting of the SiteID, the ApplicationID, and the EntityNumber. The Entity ID term shares similar heritage to the Entity Identifier record defined in DIS. (The term "Entity ID" is also sometimes used to refer to the RPR FOM 1.0 EntityType attribute - the seven unsigned fields used to indicate the specific type of entity, e.g., specific tank type, dismounted infantry, etc. - but this usage is not employed in this document.)

Message String

The client should set the message string using the following format (all values in ASCII):

```
HEADER
<attribute>=<value>
<attribute>=<value>
…
EOH
<Arbitrary data>
```

The header section starts with the string `HEADER` in the first line, followed by attribute-value pairs (each in a new line), and ends with the string `EOH` in a new line, finally followed by optional data. The possible attribute-value pairs are listed shown in Table D-6, and can occur in any order.

TABLE D-6.   Possible Attribute-Value Pairs

Name	Description	Value	Example
receiver	Destination Entity ID (only for unicast messages) optional	3 x uint16, separated by periods	receiver=1.1.1001 receiver=2.3.1005
size	Size of the message, in bytes or seconds	uint32 bytes, or float64 seconds	size=4000 bytes size=10 seconds
timeout	Number of seconds the client should wait for QualNet before discarding the message	float64	timeout=180 timeout=75
timestamp	Unique ID.	32-bit value represented in 0x12345678 hexadecimal format	timestamp=0x0A4B22E1 timestamp=0x000013DE

> **Note:**   The receiver field is optional, depending on whether the communication to be modeled is broadcast or unicast. All other fields are required.

The timestamp field can be any 32-bit value, but should be sent as a hexadecimal string. For a given ApplicationSpecificRadioSignal interaction, the same timestamp value is stored in the Timestamp field of the Process Message and Timeout interactions. In other words, this value is used to uniquely match up result interactions to the original communications modeling *request* interaction.

D.2.2 Data Interaction

QualNet uses the Data interaction to define three different types of messages:

- Process Message interaction
- Timeout interaction
- Ready To Send Signal (RTSS) interaction

The DatumID field is used to distinguish among the above message types for a given Data interaction. In all cases, the VariableDatums field should not exceed 1,000 bytes. Multiple interactions can be sent in such cases.

D.2.2.1 Process Message Interaction

QualNet sends the Process Message interaction (actually a Data interaction) when QualNet determines that one or more entities has successfully received a message. Table D-7 shows the values set by QualNet.

TABLE D-7.   Process Message Interaction Values

Field	Value				
OriginatingEntity	The Entity ID of the entity sending the radio signal.				
ReceivingEntity	Nothing				
RequestIdentifier	Nothing				
FixedDatums	Nothing				
VariableDatumSet	**Field**	**Value**			
	NumberOfVariableDatums	1			
	VariableDatums	**Field**	**Value**		
		DatumID	60,001		
		DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.		
		DatumValue	**Field**	**Value**	**Size**
			Source Entity ID	Entity ID (previously received from client)	6 bytes
			Transmitter radio index	uint16 (previously received from client)	2 bytes
			Timestamp	32-bit arbitrary data type (previously received from client)	4 bytes
			Number of receiving entities	uint32	4 bytes
			Entity 1 ID	Entity ID	6 bytes
			Entity 1 Delay	float64	8 bytes
			Entity 2 ID	Entity ID	6 bytes
			Entity 2 Delay	float64	8 bytes
			...	...	...
		PaddingTo64	Padding to 64 bits		

Note: A positive delay represents the time it took, in seconds, for the message since its transmission to reach its recipient. A negative delay value means the message is undeliverable (this is currently not supported but may be at a future date; instead, the lack of a Process Message interaction for one or more destinations by the time a Timeout interaction is issued, should be interpreted by the client as a failure for those destinations).

D.2.2.2 Timeout Interaction

QualNet sends the Timeout interaction (actually a Data interaction) when the processing of a message is complete (no further Process Message interactions will be sent for a specific request after the Timeout interaction is sent). If a Process Message interaction has not been sent for a given destination by the time the Timeout interaction is issued, the client should assume the message could not be delivered to that destination. Table D-8 shows the values set by QualNet.

TABLE D-8.   Timeout Interaction Values

Field	Value		
OriginatingEntity	The Entity ID of the entity sending the radio signal.		
ReceivingEntity	Nothing		
RequestIdentifier	Nothing		
FixedDatums	Nothing		
VariableDatumSet	**Field**	**Value**	
	NumberOfVariableDatums	1	
	VariableDatums	**Field**	**Value**
		DatumID	60,002
		DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.
		DatumValue	**Field** / **Value** / **Size** table below
		PaddingTo64	Padding to 64 bits

DatumValue table:

Field	Value	Size
Source Entity ID	Entity ID (previously received from client)	6 bytes
Radio index	uint16 (previously received from client)	2 bytes
Timestamp	32-bit arbitrary data type (previously received from client)	4 bytes
Number of packets	uint32	4 bytes
Number of receiving entities	uin32	4 bytes
Entity 1 ID	Entity ID	6 bytes
Entity 1 Status	bool8	1 byte
Entity 2 ID	Entity ID	6 bytes
Entity 2 Status	bool8	1 byte
...	...	...

Note: A status with a value of 0 means FALSE. All other status values mean TRUE.

D.2.2.3 Ready to Send Signal (RTSS) Interaction

QualNet sends the Ready To Send Signal (RTSS) interaction (actually a Data interaction) to indicate to the client that it can send an ApplicationSpecificRadioSignal interaction for a specific network. The RTSS interaction is used optionally depending on the QualNet scenario configuration. Table D-9 shows the RTSS interaction values.

TABLE D-9.   Ready to Send Signal (RTSS) Interaction Values

Field	Value			
OriginatingEntity	The Entity ID of the entity sending the radio signal			
ReceivingEntity	Nothing			
RequestIdentifier	Nothing			
FixedDatums	Nothing			
VariableDatumSet	**Field**	**Value**		
	NumberOfVariableDatums	1		
	Padding	Padding to 64 bits		
	VariableDatums	**Field**	**Value**	
		DatumID	60,010	
		DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.	
		DatumValue	**Field**	**Value**	**Size**
			Source Entity ID	Entity ID	6 bytes
			Source Radio index	uint16	2 bytes
			Timestamp	unit32	4 bytes
			Window Time	(currently unused)	4 bytes
		PaddingTo64	Padding to 64 bits	

The Timestamp field above is an actual HLA timestamp value, and does not correspond with the timestamp and Timestamp fields used in all of the other interactions.

D.3 HLA Protocol Dynamic Statistics Interface Communications Document (ICD)

The purpose of the HLA dynamic statistics interface is to allow subscribing federates to receive communications statistics from QualNet in real time as they are updated during the simulation.

D.3.1 Enabling HLA Dynamic Statistics in QualNet

Dynamic statistics over HLA can be enabled only if the QualNet GUI is used. See Chapter 5 of *QualNet User's Guide* for details of enabling dynamic statistics in the GUI.

The parameters to configure HLA statistics are described in Section 2.2.2.1.

D.3.2 Comment Interaction

QualNet sends the Comment interaction to report dynamic statistics. Table D-10 lists the parameters of the Comment interaction. Note that although all three parameters are sent by QualNet, only the VariableDatumSet parameter contains useful information. (Any relevant EntityIDs are sent in the VariableDatumSet parameter.)

TABLE D-10.   Comment Interaction Parameters

Parameter Name	Description
OriginatingEntity	The HLA entity ID of the entity or application sending the interaction. Sent as all zeroes by QualNet.
ReceivingEntity	The HLA entity ID of the entity or application which is the intended recipient of the interaction. Sent as all zeroes by QualNet.
VariableDatumSet	The set of data items (types and values) associated with the interaction.

The following three types of Comment interactions are sent by QualNet:

- nodeId Description notification
- Metric Definition notification
- Metric Update notification

Each type of Comment interaction is distinguished by the DatumID field of VariableDatumSet parameter of the Comment interaction.

VariableDatumSet Structure
Table D-11 shows the format for the VariableDatumSet parameter for all dynamic statistics related Comment interactions sent by QualNet. NumberOfVariableDatums will always be 1. The DatumValue field will always be a null-terminated string (and the size of the null byte is included in DatumLength).

TABLE D-11.   VariableDatumSet Structure

Field	Value		
VariableDatumSet	**Field**	**Value**	
	NumberOfVariableDatums	1	
	VariableDatums (One VariableDatum, shown to right)	**Field**	**Value**
		DatumID	60,101 or 60,102 or 60,103
		DatumLength	Size in bits of DatumValue
		DatumValue	Null-terminated string
		PaddingTo64	Padding to DatumValue to 64 bits

Notes: **1.** The VariableDatums field will not exceed 1000 bytes

2. DatumLength is specified in bits.

3. DatumLength is the size of DatumValue, and excludes the size of the PaddingTo64 field.

D.3.3 nodeId Description Notification

The nodeId Description notification is used to convey RPR-FOM object attributes associated with specific QualNet node IDs. This notification sends the node ID to MarkingData and node ID to RadioIndex mappings from the scenario Radios file (see Section 2.2.2.1.2). nodeId Description interactions are not sent unless the parameter `VRLINK-HLA-DYNAMIC-STATISTICS-SEND-NODEID-DESCRIPTIONS` is set to `YES` (see Section 2.2.2.1).

The DatumID field for a nodeId Description notification is set to `60,101`. The following format is used for the DatumValue field:

```
DatumValue string:
nodeId "markingData" radioIndex<CRLF>
nodeId "markingData" radioIndex<CRLF>
...
```

While each field is sent in string form (ASCII) in the Comment interaction, the Type column in the table indicates how each field can be stored internally by subscribing federates. For a Type of char[n], n includes space for a terminating null (for example, if n=11, that's 10 bytes of useful data plus one null character); when n is not specified, the string length can be arbitrarily large. See Table D-12.

TABLE D-12.   nodeId Description Notification

Field	Type	Description
nodeId	uint32	Unique identifier for QualNet nodes.
markingData	char[11]	BaseEntity.PhysicalEntity.Marking.MarkingData. (It is assumed the MarkingEncodingType indicates an ASCII-encoded MarkingData.) When no valid MarkingData is available (such as when reporting statistics for a QualNet node that is not associated with any BaseEntity object), "" is printed.
radioIndex	uint16	BaseEntity.RadioTransmitter.RadioIndex.

The QualNet simulation starts with the first federation object it discovers (a PhysicalEntity or RadioTransmitter object). Immediately after this occurs, QualNet will issue one or more nodeId Description notifications. It's important that any federate monitoring for dynamic statistics be joined to the federation and subscribed to Comment interactions when this occurs, since interactions are not buffered by the RTI when time management services are not used.

D.3.4 Metric Definition Notification

The Metric Definition notification is used to define metrics sent by QualNet using a unique metric ID. Metric Definition interactions are not sent unless the parameter `VRLINK-HLA-DYNAMIC-STATISTICS-SEND-METRIC-DEFINITIONS` is set to `YES` (see Section 2.2.2.1).

> **Note:** Metric IDs may change if new dynamic metrics are added to QualNet source code using the GUI_DefineMetric() function. Developers should not rely on metric IDs remaining static between simulation executions.

The DatumID field for a Metric Definition notification is set to `60,102`. The following format is used for the DatumValue field:

```
DatumValue string:
metricId "name" layerId dataType<CRLF>
metricId "name" layerId dataType<CRLF>
...
```

Table D-13 shows the fields in the Metric Definition notifications.

TABLE D-13.   Fields in the Metric Definition Notification

Field	Type	Description
metricId	int32	Unique identifier for QualNet metrics.
name	char[ ]	Brief string description of the metric; can include spaces.
layerId	int32	QualNet layer at which the metric resides. 0 Mobility 1 Channel 2 Physical 3 Data-link/MAC 4 Network 5 Transport 6 Application 7 Routing 8 Any (currently not used)
dataType	int32	Data type of metric. 0 int32 1 double 2 uint32

Soon after the nodeId Description notifications are sent out, one or more Metric Definition notifications will be issued by QualNet. It is possible for Metric Definition notifications to occur later in the simulation, as well. QualNet may also send out duplicate definitions.

> **Note:** QualNet only issues Metric Definition notifications at the beginning of the simulation. All GUI_DefineMetric() calls must occur before the first QualNet event.

D.3.5 Metric Update Notification

The Metric Update notification is used to convey new metric values to subscribing federates.

The DatumID field for a Metric Update notification is set to `60,103`. The format of the DatumValue field depends on the value of the parameter `VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE` (see Section 2.2.2.1). If `VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE` is set to `VERBOSE`, the following format is used:

```
DatumValue string:
nodeId "markingData" entityId radioIndex metricId "name" layerId
dataType value time<CRLF>
nodeId "markingData" entityId radioIndex metricId "name" layerId
dataType value time<CRLF>
...
```

If `VRLINK-HLA-DYNAMIC-STATISTICS-METRIC-UPDATE-MODE` is set to `BRIEF`, the following format is used for the DatumValue field:

```
DatumValue string:
nodeId metricId value time<CRLF>
nodeId metricId value time<CRLF>
...
```

Many of the fields are the same as those for nodeId Description and Metric Definition notifications. The additional fields are described in Table D-14.

TABLE D-14.   Additional Fields in the Metric Update Notification

Field	Type	Description
nodeId	uint32	Unique identifier for a QualNet node.
entityId	uint16[3]	BaseEntity.EntityIdentifier for the RPR-FOM BaseEntity (and subclasses) object, if any, associated with a QualNet node. Printed as a triplet of integers delimited by periods, for example., 1.23.456 (SiteID = 1, ApplicationID = 23, EntityNumber = 456). When no valid EntityID is available (such as when reporting statistics for a QualNet node that is not associated with any BaseEntity object), this field is printed as -1 (no extra integers).
radioIndex	uint32	EmbeddedSystem.RadioTransmitter.RadioIndex for the RPR-FOM RadioTransmitter object, if any, associated with a QualNet node. When no valid RadioIndex is available, this field is printed as -1.
metricId	int32	Unique identifier for a QualNet metric.
value	Depends on Type	Metric value expressed as an ASCII string, as usual. The value can be stored according to the dataType assigned to the metric in the Metric Definition notification.
time	uint64	QualNet simulation time in units of nanoseconds (expressed as an integer in string form) at which the metric update is valid. Note that QualNet simulation time starts at zero when QualNet discovers the first PhysicalEntity or RadioTransmitter object. Note: While this value can be stored as a uint64 by subscribing federates, federates can also opt to convert it into a double, by casting to double and then dividing by 1e9 (ending up with 1.0 = 1 second of QualNet simulation time).

Notice that in Verbose mode, the client federate does not need to parse Metric Definition notifications since all the information is present in Metric Update notifications. In Brief mode, the Metric Definition notifications need to be parsed by the client federate in order to associate metric ID values with the metric name, layer, and data type; whereas.

Metric Update notifications are sent by QualNet throughout simulation execution. They are sent every 1 second of simulation time (if there is any updated metric), or when the size threshold of 1,000 bytes of VariableDatums field is exceeded.

A reporting interval is also specified in the GUI (by going to **Animation > Dynamic Statistics > Scenario Statistics** and setting the parameter **Update Interval Time**). At these intervals, the GUI_Send{Integer,Unsigned,Real}Data() functions are called; these functions load data into the buffer used for Metric Update notifications. Therefore, if the user is mainly interested in dynamic statistics over HLA (i.e., not really interested in the GUI display), the user should set a reporting interval at or around 1 second.

Notes: 1. The GUI_Send{...}Data() calls currently repeat metric values even when those values have not changed. This can be improved to minimize federation bandwidth consumption.

2. Metric Update notifications can be sent for node IDs for which a nodeId Description notification has not been sent. These correspond to QualNet nodes which do not map to any federation object (PhysicalEntity or RadioTransmitter).

D.4 DIS PDUs

In DIS, data messages are sent as Protocol Data Units (PDUs). The PDU types used by QualNet are:

- Entity State PDU: This PDU is sent by external simulators and describes the entities in the simulation exercise. QualNet does not send Entity State PDUs.

- Transmitter PDU: This PDU is sent by external simulation applications and is used to indicate the transmitter state.

- Signal PDU: This PDU is sent by external simulation applications and is used to request communications effects modeling.

- Data PDU: This PDU is sent by QualNet and used to pass the radio communication results to the requesting applications

■ ■

D.5 DIS Extensions ICD

The "client" is an external simulation application which sends DIS Signal PDUs to QualNet to request modeling of communication effects. QualNet sends DIS Data PDUs to the client to report on results of communications.

uint16 refers to a 16-bit unsigned integer; float32 refers to a 32-bit point floating point number; and so on, using the standard representations in C and C++. All fields are in network byte order (most significant byte occurring first).

The QualNet DIS protocol does not define any new PDUs, but rather uses existing PDUs containing QualNet specific data. To determine that a PDU is specific to itself, QualNet uses the UserProtocolID parameter for Signal PDUs with Application-Specific Data, and the DatumID field for Data PDUs.

It is assumed that the client will create the necessary object instances that correspond to the simulation entities sending and receiving communications.

D.5.1 Signal PDU

The client sends the Signal PDU to request communication effects modeling from QualNet. The interaction parameters should have the values shown in Table D-5.

TABLE D-15.   Signal PDU

Field	Value	
PDU Header	As defined by the DIS Standard	
Entity ID	Entity ID of the source entity	
Radio ID	Radio index of the source radio	
Encoding Scheme Record	**Field**	**Value**
	Encoding Class	Application-specific Data (2)
	Encoding Type	Not used
Sample Rate	Requested data rate	
Data length	Number of bytes in the message string	
Samples	**Field**	**Value**
	UserProtocolID	10000
	Message String	Text

Note: QualNet uses the UserProtocolID parameter to recognize that the PDU is specific to QualNet and will ignore all other Signal PDUs.

Message String

The client should set the message string using the following format (all values in ASCII):

```
HEADER
<attribute>=<value>
<attribute>=<value>
…
EOH
<Arbitrary data>
```

The header section starts with the string HEADER in the first line, followed by attribute-value pairs (each in a new line), and ends with the string EOH in a new line, finally followed by optional data. The possible attribute-value pairs are listed shown in Table D-16, and can occur in any order.

TABLE D-16.   Possible Attribute-Value Pairs

Name	Description	Value	Example
receiver	Destination Entity ID (only for unicast messages) optional	3 x uint16, separated by periods	receiver=1.1.1001 receiver=2.3.1005
size	Size of the message, in bytes or seconds	uint32 bytes, or float64 seconds	size=4000 bytes size=10 seconds
timeout	Number of seconds the client should wait for QualNet before discarding the message	float64	timeout=180 timeout=75
timestamp	Unique ID.	32-bit value represented in 0x12345678 hexadecimal format	timestamp=0x0A4B22E1 timestamp=0x000013DE

> **Note:**   The receiver field is optional, depending on whether the communication to be modeled is broadcast or unicast. All other fields are required.

The timestamp field can be any 32-bit value, but should be sent as a hexadecimal string. For a given Signal PDU, the same timestamp value is stored in the Timestamp field of the Process Message and Timeout results. In other words, this value is used to uniquely match up results to the original communications modeling *request* PDU.

D.5.2 Data PDU

QualNet uses the Data PDU to define three different types of messages:

- Process Message result
- Timeout result
- Ready To Send Signal (RTSS) result

The DatumID field is used to distinguish among the above message types. In all cases, the VariableDatums field should not exceed 1,000 bytes. Multiple PDUs can be sent in such cases.

D.5.2.1 Process Message Result

QualNet sends the Process Message result (actually a Data PDU) when QualNet determines that one or more entities has successfully received a message. Table D-17 shows the values set by QualNet.

TABLE D-17.   Data PDU for Process Message Result

Field	Value		
PDU Header	As defined by the DIS standard.		
OriginatingEntity ID	The Entity ID of the entity sending the radio signal.		
ReceivingEntity ID	Nothing		
Request ID	Nothing		
Number of Fixed Data Records	0		
FixedDatums	Not Fixed Datums		
NumberofVariable Dataums	1		
VariableDatums	**Field**	**Value**		
	DatumID	60,001		
	DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.		
	DatumValue	**Field**	**Value**	**Size**
		Source Entity ID	Entity ID (previously received from client)	6 bytes
		Transmitter radio index	uint16 (previously received from client)	2 bytes
		Timestamp	32-bit arbitrary data type (previously received from client)	4 bytes
		Number of receiving entities	uint32	4 bytes
		Entity 1 ID	Entity ID	6 bytes
		Entity 1 Delay	float64	8 bytes
		Entity 2 ID	Entity ID	6 bytes
		Entity 2 Delay	float64	8 bytes
		...	...	...
		PaddingTo64	Padding to 64 bits	

Note: A positive delay represents the time it took, in seconds, for the message since its transmission to reach its recipient. A negative delay value means the message is undeliverable (this is currently not supported; instead, the lack of a Process Message result for one or more destinations by the time a Timeout result is issued, should be interpreted by the client as a failure for those destinations).

D.5.2.2 Timeout Result

QualNet sends the Timeout result (actually a Data PDU) when the processing of a message is complete (no further Process Message results will be sent for a specific request after the Timeout result is sent). If a Process Message result has not been sent for a given destination by the time the Timeout result is issued, the client should assume the message could not be delivered to that destination. Table D-18 shows the values set by QualNet.

TABLE D-18.   Data PDU for Timeout Result

Field	Value		
PDU Header	As defined by the DIS standard.		
OriginatingEntity ID	The Entity ID of the entity sending the radio signal.		
ReceivingEntity ID	Nothing		
Request ID	Nothing		
Number of Fixed Data Records	0		
FixedDatums	Not Fixed Datums		
NumberofVariable Dataums	1		
VariableDatums	**Field**	**Value**	
	DatumID	60,002	
	DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.	
	DatumValue	**Field**	**Value**	**Size**
		Source Entity ID	Entity ID (previously received from client)	6 bytes
		Transmitter radio index	uint16 (previously received from client)	2 bytes
		Timestamp	32-bit arbitrary data type (previously received from client)	4 bytes
		Number of packets	uint32	4 bytes
		Number of receiving entities	uint32	4 bytes
		Entity 1 ID	Entity ID	6 bytes
		Entity 1 Status	Bool8	1 byte
		Entity 2 ID	Entity ID	6 bytes
		Entity 2 Status	Bool8	1 byte
		...	...	...
		PaddingTo64	Padding to 64 bits	

Note: A status with a value of 0 means FALSE. All other status values mean TRUE.

D.5.2.3 Ready to Send Signal (RTSS) Result

QualNet sends the Ready To Send Signal (RTSS) result (actually a Data PDU) to indicate to the client that it can send a Signal PDU for a specific network. The RTSS result is used optionally depending on the QualNet scenario configuration. Table D-19 shows the RTSS result values.

TABLE D-19.   Data PDU for RTSS Result

Field	Value		
PDU Header	As defined by the DIS standard.		
OriginatingEntity ID	The Entity ID of the entity sending the radio signal.		
ReceivingEntity ID	Nothing		
Request ID	Nothing		
Number of Fixed Data Records	0		
FixedDatums	Not Fixed Datums		
NumberofVariable Dataums	1		
VariableDatums	**Field**	**Value**	
	DatumID	60,0102	
	DatumLength	Length in bits of the DatumValue field, excluding PaddingTo64.	
	DatumValue	**Field**	**Value**	**Size**
		Source Entity ID	Entity ID	6 bytes
		Source radio index	uint16	2 bytes
		Timestamp	unit32	4 bytes
		Window Time	(Currently not used)	4 bytes
		PaddingTo64	Padding to 64 bits	

The Timestamp field above is an actual DIS timestamp value, and does not correspond with the timestamp and Timestamp fields used in all of the other PDUs.

E MTS Emulator

To test the functionality of the Socket Interface, a program called MTS was developed at Scalable Network Technologies. MTS is a fast and light-weight stand-alone program whose only purpose is to send messages to the QualNet and to receive responses over the Socket Interface. This section describes the details of the MTS program.

The QualNet Socket and MTS program interact through a TCP/IP socket. The default port is 5033, but the port can be changed in the QualNet configuration file by means of the `SOCKET-INTERFACE-PORT` parameter (see Section 3.3). The configuration files for QualNet and MTS determine their behavior and the messages exchanged between them.

This section is organized as follows:

- Section E.1 describes the commands that can be included in the MTS configuration file.
- Section E.2 describes the steps to run QualNet with MTS.
- Section E.3 describes a sample scenario and the input and output files for that scenario.

E.1 MTS Configuration File

The MTS configuration file determines the actions of MTS, such as times of node creation, mobility of nodes, when communications occur, and when to advance time. The MTS configuration file is similar in style to the QualNet scenario configuration file in that:

- Commands may be put in the file in any order.
- Each command is on one line.
- Lines may be commented with the '#' character.
- Each command consists of a command name followed by a list of required parameters and a list (possibly empty) of optional parameters. The required parameters must appear in the order specified. Optional parameters can appear in any order after the required parameters.

> **Note:** Due to the limitation of the page width in this document, some commands are shown to span several lines. However, in the configuration files, there should not be any line breaks in the middle of a command.

The MTS commands and their associated parameters are listed below.

- **TimeManagementMode**: This command specifies the execution mode of MTS. The execution mode can be real time mode, time managed mode, or time managed fast mode.
  - In real time mode, MTS and QualNet run at wall-clock speed.
  - In time managed mode, MTS runs at a specified speed. The speed is specified as a multiple of wall-clock speed.
  - In time managed fast mode, MTS runs as fast as possible with the specified look-ahead interval.

  The syntax of the TimeManagementMode command is:

  ```
  TimeManagementMode <mode>
  ```

  The parameter is described in Table E-1.

  > **Note:** If more than one TimeManagementMode command is included in the MTS configuration file, the last TimeManagementMode command in the file determines the execution mode

TABLE E-1.   Parameters of TimeManagementMode Command

Command Parameter	Options	Description
`<mode>` *Required*		Execution mode of QualNet and MTS.
	• `RealTime`	If the value is `RealTime`, MTS runs in real time mode.
	• `TimeManaged [<speed>]`	If the value is `TimeManaged [<speed>]`, MTS runs in time managed mode at `<speed>` times wall-clock speed. `<speed>` is specified as a real number. If `<speed>` is not specified, the default speed-up factor is 2.0.
	• `TimeMangedFast [<interval>]`	If the value is `TimeManagedFast [<interval>]`, MTS runs in time managed fast mode with a look-ahead interval of `<interval>` seconds. `<interval>` is specified as a real number. If `<interval>` is not specified, the default look-ahead interval is 1.0 second.

Examples of the TimeManagementMode command are:

```
TimeManagementMode RealTime
TimeManagementMode TimeManaged 3.0
TimeManagementMode TimeManagedFast 2.0
```

- **Address**: This command specifies the IP address to which MTS is to connect.

  The syntax of the Address command is:

  ```
  Address <address>
  ```

  The parameter is described in Table E-2. If this command not included in the MTS configuration file, the default IP address to connect to is 127.0.0.1.

  TABLE E-2.   Parameters of Address Command

Command Parameter	Type/Options	Description
<address> *Required*	IP Address	The IP address to which MTS is to connect.

- **Port**: This command specifies the port number to which MTS is to connect.

  The syntax of the Port command is:

  ```
  Port <port>
  ```

  The parameter is described in Table E-3. If this command not included in the MTS configuration file, the default port to connect to is 5033.

  TABLE E-3.   Parameters of Port Command

Command Parameter	Type/Options	Description
<port> *Required*	Integer	The port number to which MTS is to connect.

- **PrintTimeAdvance**: This command causes MTS to print time advance information when running in time managed and time managed fast modes (see Table E-1).

  The syntax of the PrintTimeAdvance command is:

  ```
  PrintTimeAdvance
  ```

  This command does not have any parameters.

- **InitializeSimulation**: This command begins a QualNet simulation. QualNet does not begin the simulation until the InitializeSimulation command is received. This message is required for both time managed and real time modes.

  The syntax of the InitializeSimulation command is:

  ```
  InitializeSimulation [COORDINATESYSTEM <value>]
                       [SCENARIO <scenariostring>]
  ```

  The parameters are described in Table E-4.

TABLE E-4.   Parameters of InitializeSimulation Command

Command Parameter	Type/Options	Description
COORDINATESYSTEM <value> *Optional*	• Cartesian • LatLonAlt • GCCCartesian	String indicating the coordinate system to be used. If this parameter is not specified, the Cartesian system is used by default.
SCENARIO <scenariostring> *Optional*	LongString	Description of the scenario.

> **Note:** The value of the parameter `COORDINATE-SYSTEM` in the QualNet configuration file should be the same as the value of the parameter `COORDINATESYSTEM` of the InitializeSimulation command. If the values are different, an error message will be sent back to the external program and QualNet will not be initialized.

- **TestWarmup**: This command causes MTS to test the QualNet Warm-up phase (see Section 3.2.3). If TestWarmup is enabled, MTS will send all messages that are scheduled for 0 seconds to QualNet. `CreatePlatform` messages that will be active during the Warm-up phase should be scheduled for 0 seconds. Then it will send a `BeginWarmup` message and wait for QualNet to enter the Initialized state. Additionally, it will send several `CommEffectsRequest` messages.

  The syntax of the TestWarmup command is:

  ```
  TestWarmup
  ```

  This command does not have any parameters.

- **CreatePlatform**: This command causes a `CreatePlatform` message to be sent once, or repeatedly after a fixed interval. The `CreatePlatform` message creates an entity at the specified location.

  The syntax of the CreatePlatform command is:

  ```
  CreatePlatform <entityID> <time> [X <x-value>] [Y <y-value>]
                 [Z <z-value>] [VELX <x-value>] [VELY <y-value>]
                 [VELZ <z-value>] [DAMAGESTATE <state>]
                 [TYPE <platformtype>] [MULTICASTGROUPS <group>]
  ```

  or
  ```
  CreatePlatform <entityID> <time> [LAT <lat-value>] [LON <lon-value>]
                 [ALT <alt-value>] [VELLAT <lat-value>]
                 [VELLON <lon-value>] [VELALT <alt-value>]
                 [DAMAGESTATE <state>] [TYPE <platformtype>]
                 [MULTICASTGROUPS <group>]
  ```

  The parameters are described in Table E-5.

TABLE E-5.   Parameters of CreatePlatform Command

Command Parameter	Type/Options	Description
`<entityID>` *Required*	String	Identifier of the entity to be created.
`<time>` Required	Real	Time (in seconds) when to send the first `CreatePlatform` message.
`X <x-value>` *Optional*	Real	x-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the x-coordinate is 0.
`Y <y-value>` *Optional*	Real	y-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the y-coordinate is 0.
`Z <z-value>` *Optional*	Real	z-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the z-coordinate is 0.
`LAT <lat-value>` *Optional*	Real	Latitude (in degrees). If the Lat/Lon/Alt system is used and this parameter is not specified, the latitude is 0.
`LON <lon-value>` *Optional*	Real	Longitude (in degrees). If the Lat/Lon/Alt system is used and this parameter is not specified, the longitude is 0.
`ALT <alt-value>` *Optional*	Real	Altitude (in meters). If the Lat/Lon/Alt system is used and this parameter is not specified, the altitude is 0.
`VELX <x-value>` *Optional*	Real	Velocity (in meters/sec) of the entity in the X direction.
`VELY <y-value>` *Optional*	Real	Velocity (in meters/sec) of the entity in the Y direction.

TABLE E-5.   Parameters of CreatePlatform Command (Continued)

Command Parameter	Type/Options	Description
VELZ <z-value> *Optional*	Real	Velocity (in meters/sec) of the entity in the Z direction.
VELLAT <lat-value> *Optional*	Real	Velocity (in degrees/sec) of the entity along the latitude.
VELLON <lon-value> *Optional*	Real	Velocity (in degrees/sec) of the entity along the longitude.
VELALT <alt-value> *Optional*	Real	Velocity (in meters/sec) of the entity along the altitude.
DAMAGESTATE <state> *Optional*	• 0 • 1	Bitwise kill state. 0: Communications are not damaged 1: Communications are damaged
TYPE <platformtype> *Optional*	• 0 • 1	Platform Type. 0: Ground node 1: Air node If this parameter is not specified, the node is a ground node.
MULTICASTGROUPS <group> *Optional*	ListofGroups	A list of multicast groups that this platform will join.

- **CommEffectsRequest**: This command causes a CommEffectsRequest message to be sent once or repeatedly after a fixed interval.

  The syntax of the CommEffectsRequest command is:

```
CommEffectsRequest <src> <dst> <time> [INTERVAL <interval>]
                    [ID1 <value>] [ID2 <value>] [<transport-protocol>]
                    [SIZE <size>] [DESCRIPTION <desc>]
                    [FAILURETIMEOUT <timeout>]
                    [PRECEDENCE <precedence>] [DSCP <value>] [TOS <tos>]
                              [TTL <ttl>]  [END-TIME <end-time>]
```

Note: A CommEffectsRequest command can include at most one of the three parameters: PRECEDENCE, DSCP, and TOS.

The parameters are described in Table E-6.

TABLE E-6.   Parameters of CommEffectsRequest Command

Command Parameter	Type/Options	Description
`<src>` *Required*	String	Identifier of the source entity.
`<dst>` *Required*	String	Identifier of the destination entity.
`<time>` *Required*	Real	Time (in seconds) when to send the first `CommEffectsRequest` message.
`INTERVAL <interval>` *Optional*	Real	Time to wait (in seconds) before resending the `CommEffectsRequest` message. If this parameter is not specified, the `CommEffectsRequest` message is sent only once.
`ID1 <value>` *Optional*	Integer	First 8 bits of the message identification field. If this parameter is not specified, the `ID1` is set to 0.
`ID2 <value>` *Optional*	Integer	Second 8 bits of the message identification field. If this parameter is not specified, the `ID2` is set to 0.
`<transport-protocol>` *Optional*	• `TCP` • `UDP`	Transport protocol to use (TCP or UDP). If this parameter is not specified, the UDP protocol is used.
`SIZE <size>` *Optional*	Integer	Message size (in bytes). If this parameter is not specified, the default size is 128 bytes.
`DESCRIPTION <desc>` *Optional*	String	Description of the `CommEffectsRequest` message. This is not used by QualNet but is included to let MTS provide a description for each `CommEffectsRequest` message.
`FAILURETIMEOUT <timeout>` *Optional*	Real	Message failure timeout value (in seconds). If this parameter is not specified, then the timeout value specified in the configuration file (`SOCKET-INTERFACE-UDP-FAILURE-TIMEOUT` or `SOCKET-INTERFACE-TCP-FAILURE-TIMEOUT`) is used.

`PRECEDENCE <precedence>`
 	
Optional
	String	Message precedence.
		
		The following values can be used to indicate the precedence level:
		
		`ROUTINE`
		`PRIORITY`
		`IMMEDIATE`
		`FLASH`
		`FLASHOVERRIDE`
		`CRITICAL`
		`INTERNETCONTROL`
		`NETCONTROL`

TABLE E-6.   Parameters of CommEffectsRequest Command (Continued)

Command Parameter	Type/Options	Description
DSCP <value>\n\n*Optional*	Integer	Request DSCP.\n\nThe valid range for this field is 0-63.\n\nThe bits indicate the following:\n\n 0-2: Precedence\n\n 3: Low Delay\n\n 4: High Throughput\n\n 5: High Reliability\n\nFor example, if the DSCP field is set to 10 (001010 in binary), then the message is given a precedence value of 1 with High Throughput.
TOS <tos>\n\n*Optional*	Integer	Request TOS.\n\nThe valid range for this field is 0-255 (i.e., all 8 bits can be used).\n\nThis value replaces the entire TOS byte in the IP header.
TTL <ttl>\nOptional	Real	Time-To-Live (TTL) field for communications (in seconds).\n\nThis is valid for TCP and UDP communications. For TCP communications, the first TCP packet sent from the source to the destination should specify the TTL that should be used for all subsequent packets from the source to the destination.
END-TIME <end-time>\nOptional	Real	Time to stop sending the CommEffectsRequest message (in seconds).\n\nIf this parameter is not specified, the CommEffectsRequest message is sent continually until the end of the simulation.

- **PrintSendMessage**: This command causes MTS to print information about CommEfectsRequest and AdvanceSimulationTime messages, in addition to the responses.

  The syntax of the PrintSendMessage command is:

  ```
  PrintSendMessage
  ```

  This command does not have any parameters.

• **UpdatePlatform**: This command causes an `UpdatePlatform` message to be sent once or repeatedly after a fixed interval. The `UpdatePlatform` message updates the position of an entity to the specified coordinates.

The syntax of the UpdatePlatform command is:

```
UpdatePlatform <entityID> <time> [X <x-value>] [Y <y-value>]
               [Z <z-value>] [VELX <x-value>] [VELY <y-value>]
               [VELZ <z-value>] [DAMAGESTATE <state>]
               [JOINMULTICASTGROUPS <join-group>]
               [LEAVEMULTICASTGROUP <leave-group>]
```

or

```
UpdatePlatform <entityID> <time> [LAT <lat-value>] [LON <lon-value>]
               [ALT <alt-value>] [VELLAT <lat-value>]
               [VELLON <lon-value>] [VELALT <alt-value>]
               [DAMAGESTATE <state>]
               [JOINMULTICASTGROUPS <join-group>]
               [LEAVEMULTICASTGROUP <leave-group>]
```

Note: If the GCC-Cartesian coordinate system is used, then the X, Y, and Z values must be specified for position update. If these values are not specified, position update will not be sent to QualNet.

If the Cartesian coordinate system is used, then the X and Y values must be specified for position update. If these values are not specified, position update will not be sent to QualNet.

If the Lat/Lon/Alt coordinate system is used, then the LAT and LON values must be specified for position update. If these values are not specified, position update will not be sent to QualNet.

The parameters are described in Table E-7.

TABLE E-7.   Parameters of UpdatePlatform Command

Command Parameter	Type/Options	Description
`<entityID>` *Required*	String	Identifier of the entity to be updated.
`<time>` *Required*	Real	Time (in seconds) when to send the first `UpdatePlatformState` message.
`X <x-value>` *Optional*	Real	New x-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the x-coordinate is not updated.
`Y <y-value>` *Optional*	Real	New y-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the y-coordinate is not updated.
`Z <z-value>` *Optional*	Real	New z-coordinate (in meters) of the entity. If the Cartesian coordinate system is used and this parameter is not specified, the z-coordinate is not updated.

TABLE E-7.   Parameters of UpdatePlatform Command (Continued)

Command Parameter	Type/Options	Description
LAT <lat-value> Optional	Real	New latitude (in degrees). If the Lat/Lon/Alt system is used and this parameter is not specified, the latitude is not updated.
LON <lon-value> *Optional*	Real	New longitude (in degrees). If the Lat/Lon/Alt system is used and this parameter is not specified, the longitude is not updated.
ALT <alt-value> *Optional*	Real	New altitude (in meters). If the Lat/Lon/Alt system is used and this parameter is not specified, the altitude is not updated.
VELX <x-value> *Optional*	Real	Velocity (in meters/sec) of the entity in the X direction.
VELY <y-value> *Optional*	Real	Velocity (in meters/sec) of the entity in the Y direction.
VELZ <z-value> *Optional*	Real	Velocity (in meters/sec) of the entity in the Z direction.
VELLAT <lat-value> *Optional*	Real	Velocity (in degrees/sec) of the entity along the latitude.
VELLON <lon-value> *Optional*	Real	Velocity (in degrees/sec) of the entity along the longitude.
VELALT <alt-value> *Optional*	Real	Velocity (in meters/sec) of the entity along the altitude.

DAMAGESTATE <state>
 	
Optional
	• 0
	• 1
		Bitwise kill state.
		
		    0: Communications are not damaged
		    1: Communications are damaged

JOINMULTICASTGROUPS <join-group> *Optional*	ListofGroups	A list of multicast groups to join.
LEAVEMULTICASTGROUPS <leave-group> *Optional*	ListofGroups	A list of multicast groups to leave.

- **TimeStamp:** This command enables the use of timestamps in a scenario.
  The syntax of the command is:

  ```
  TimeStamp YES | NO
  ```

  If the argument is YES, then timestamps are enabled; otherwise, timestamps are not enabled. If this command is not included in the MTS configuration file, default values are used. By default, timestamps are enabled for time managed mode and are not enabled for real time mode.

- **PauseSimulation:** This command causes a `PauseSimulation` message to be sent at the specified time. This results in QualNet moving to the Paused state from the Initialized state or Executing state.

  The syntax of the PauseSimulation command is:

  ```
  PauseSimulation [<time>] [DURATION <duration>]
  ```

  The parameters are described in Table E-8.

TABLE E-8.   Parameters of PauseSimulation Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `PauseSimulation` message. If this parameter is not specified, the default value is 0.0.
`DURATION <duration>` *Optional*	Real	Duration (in seconds) of the pause. If this parameter is not specified, the default value is 0.0.

- **StopSimulation:** This command causes a `StopSimulation` message to be sent at the specified time. This results in gracefully ending the QualNet simulation.

  The syntax of the StopSimulation command is:

  ```
  StopSimulation [<time>]
  ```

  The parameters are described in Table E-9.

TABLE E-9.   Parameters of StopSimulation Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `StopSimulation` message. If this parameter is not specified, the default value is 0.0.

- **ResetSimulation:** This command causes a `ResetSimulation` message to be sent at the specified time. This results in QualNet internal data structures being cleaned and QualNet returning to the standby state.

  This gracefully ends the QualNet simulation like the StopSimulation command, however, in addition, the QualNet socket connections are closed and statistics are output to the statistics log file.

  The syntax of the ResetSimulation command is:

  ```
  ResetSimulation [<time>]
  ```

  The parameters are described in Table E-10.

TABLE E-10.   Parameters of ResetSimulation Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `ResetSimulation` message. If this parameter is not specified, the default value is 0.0.

- **Read:** This dynamic command causes a `Read` message to be sent at the specified time. This allows scenario parameters to be queried during simulation.

  The syntax of the Read command is:

  ```
  Read [<time>] [PATH <path>]
  ```

  The parameters are described in Table E-11.

TABLE E-11.   Parameters of Read Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `Read` message. If this parameter is not specified, the default value is 0.0.
`PATH <path>` *Optional*	String	Path of the dynamic object in the dynamic hierarchy to perform the operation on.

- **Write:** This dynamic command causes a `Write` message to be sent at the specified time. This allows scenario parameters to be modified during simulation.

  The syntax of the Write command is:

  ```
  Write [<time>] [PATH <path>] [ARGS <arguments>]
  ```

  The parameters are described in Table E-12.

TABLE E-12.   Parameters of Write Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `Write` message. If this parameter is not specified, the default value is 0.0.
`PATH <path>` *Optional*	String	Path of the dynamic object in the dynamic hierarchy to perform the operation on.
`ARGS <arguments>` *Optional*	String	Arguments to write to the dynamic object.

- **Execute:** This dynamic command causes an `Execute` message to be sent at the specified time. This allows scenario parameters to be modified during simulation.

  The syntax of the Execute command is:

  ```
  Execute [<time>] [PATH <path>] [ARGS <arguments>]
  ```

  The parameters are described in Table E-13.

TABLE E-13.   Parameters of Execute Command

Command Parameter	Type/Options	Description
`<time>` *Required*	Real	Time (in seconds) to send the `Execute` message. If this parameter is not specified, the default value is 0.0.
`PATH <path>` *Optional*	String	Path of the dynamic object in the dynamic hierarchy to perform the operation on.
`ARGS <arguments>` *Optional*	String	Arguments to write to the dynamic object.

- **GetRequest:** This SNMP command causes a `GetRequest` message to be sent at the specified time. This allows scenario parameters to be queried during simulation.

  The syntax of the GetRequest command is:

  ```
  GetRequest <entity-id> [<time>] [OID <oid>]
  ```

  The parameters are described in Table E-14.

TABLE E-14.   Parameters of GetRequest Command

Command Parameter	Type/Options	Description
`<entity-id>` *Required*	String	Identification of the entity on which to perform the operation.
`<time>` *Optional*	Real	Time (in seconds) to send the `GetRequest` message. If this parameter is not specified, the default value is 0.0.
`OID <oid>` *Optional*	ListofString	List of object identifiers on which to perform the operation.

- **GetNextRequest:** This SNMP command causes a `GetNextRequest` message to be sent at the specified time. This allows scenario parameters to be queried during simulation.

  The syntax of the GetNextRequest command is:

  ```
  GetNextRequest <entity-id> [<time>] [OID <oid>]
  ```

  The parameters are described in Table E-15.

TABLE E-15.   Parameters of GetNextRequest Command

Command Parameter	Type/Options	Description
`<entity-id>` *Required*	String	Identification of the entity on which to perform the operation.
`<time>` *Optional*	Real	Time (in seconds) to send the `GetRequest` message. If this parameter is not specified, the default value is 0.0.
`OID <oid>` *Optional*	ListofString	List of object identifiers on which to perform the operation.

- **SetRequest:** This SNMP command causes a `SetRequest` message to be sent at the specified time. This allows scenario parameters to be modified during simulation.

    The syntax of the SetRequest command is:

        SetRequest <entity-id> [<time>] [OID <oid>] [VALUE <value>]

    The parameters are described in Table E-16.

TABLE E-16.   Parameters of SetRequest Command

Command Parameter	Type/Options	Description
`<entity-id>` *Required*	String	Identification of the entity on which to perform the operation.
`<time>` *Optional*	Real	Time (in seconds) to send the `SetRequest` message. If this parameter is not specified, the default value is 0.0.
`OID <oid>` *Optional*	ListofString	List of object identifiers on which to perform the operation.
`VALUE <value>` *Optional*	ListofString	List of new values for MIBS objects on which the operation is performed.

- **GetBulkRequest:** This SNMP command causes a `GetBulkRequest` message to be sent at the specified time. This allows scenario parameters to be queried during simulation.

    The syntax of the GetBulkRequest command is:

        GetBulkRequest <entity-id> [<time>] [NUMCSCALAR <num-scalar>]
                       [MAXREPEAT <max-repeat>] [OID <oid>]

    The parameters are described in Table E-17.

TABLE E-17.   Parameters of GetBulkRequest Command

Command Parameter	Type/Options	Description
`<entity-id>` *Required*	String	Identification of the entity on which to perform the operation.
`<time>` *Optional*	Real	Time (in seconds) to send the `GetBulkRequest` message. If this parameter is not specified, the default value is 0.0.
`NUMSCALAR <num-scalar>` *Optional*	UInt16	Number of scalar MIBS objects.
`MAXREPEAT <max-repeat>` *Optional*	UInt16	Maximum repetitions of a non-scalar MIBS object.
`OID <oid>` *Optional*	ListofString	List of object identifiers on which to perform the operation.

- **QuerySimulationState:** This command causes a `QuerySimulationState` message to be sent at the specified time. This allows the simulation state to be queried during the simulation.

  The syntax of the QuerySimulationState command is:

  ```
  QuerySimulationState [<time>]
  ```

  The parameters are described in Table E-18.

TABLE E-18.   Parameters of QuerySimulationState Command

Command Parameter	Type/Options	Description
`<time>` *Optional*	Real	Time (in seconds) to send the `QuerySimulationState` message. If this parameter is not specified, the default value is 0.0.

. . . . .

E.2 Running QualNet and MTS

To run QualNet with MTS using the Socket Interface, perform the following steps:

1. Set up a scenario by editing the QualNet configuration file (see Section 3.3.1) and the MTS configuration file (see Section E.1).
2. Open two command windows. In one window, change the directory to the one where the QualNet configuration file is stored. In the other window, change the directory to the one where the MTS configuration file is stored.
3. In the QualNet command window, type the following command:

   For Windows:

   ```
   %QUALNET_HOME%\bin\qualnet <QualNet-config-file>
   ```

   For Linux:

   ```
   $QUALNET_HOME/bin/qualnet <QualNet-config-file>
   ```

   where

   `<QualNet-config-file>`   Name of the QualNet configuration file (see Section 3.3.1).

4. In the MTS window, type the following command:

For Windows:

```
%QUALNET_HOME%\bin\mts-socket <MTS-configuration-file>
```

For Linux:

```
$QUALNET_HOME/bin/mts-socket <MTS-configuration-file>
```

where

`<MTS-configuration-file>`   Name of the MTS configuration file (see Section E.1).

．．．

E.3 Sample Scenario for Running QualNet and MTS

This section describes a sample scenario for running QualNet with MTS using the Socket Interface. Section E.3.1 gives a description of the scenario to be simulated. Section E.3.2 describes the input files (MTS configuration file, QualNet configuration file, and the entity mapping file) for the scenario. Section E.3.3 describes the output (log) files produced for the scenario.

E.3.1 Scenario Description

The following is a description of the sample scenario.

1. QualNet can simulate 5.0 seconds ahead of MTS.
2. Three nodes, node A100, node A200, and node A300, are created at time 10. Nodes A100 and A200 are ground nodes with initial position (41.000005, 46.000005). Node A300 is an air node with initial position (41.000005, 46.000005, 5.0).
3. Starting at 150 seconds, node 1 sends 512-byte TCP messages to node 2 every 2 seconds.
4. Starting at 150 seconds, node 2 sends 512-byte TCP messages to node 1 every 2 seconds.
5. Node A200 is damaged at 250 seconds.

E.3.2 Input Files

In this section, we describe how to set up the input files to run the scenario described in Section E.3.1.

E.3.2.1 MTS Configuration File

The MTS configuration file for the sample scenario of Section E.3.1 is called mts-config.txt and is shown in Figure E-1. Comments in the file explain how each command corresponds to the scenario.

Notice that two extra commands, which are extraneous to the scenario Section E.3.1, are included in the configuration file. These commands are included to illustrate the type of error and response messages generated. The first of these is a command to change a node's position beyond the range and results in ana error message (see Section E.3.3.2). The other is a communications request to a damaged node and results in a communications response indicating a failure (see Section E.3.3.4).

  Note: Commands can be entered in the file in any order.

```
# File name: mts-config.txt

# Tell MTS which IP address to connect to.
Address 127.0.0.1

# Tell MTS which port to connect to.
Port 5032

# Initializes simulation and set the coordinate system.
InitializeSimulation COORDINATESYSTEM LatLonAlt

# Tell QualNet which execution mode to use, and how far ahead
# it can simulate.
TimeManagementMode TimeManagedFast 5.0

# Create A100 (ground node) at location (41.000005, 46.000005, 0.0) at time 10.
CreatePlatform A100 10.0 X 41.00005 Y 46.000005 Z 0.0  TYPE 0

# Create A200 (ground node by default) at location (41.000005, 46.000005, 0.0)
# at time 10.
CreatePlatform A200 10.0 X 41.00005 Y 46.000005 Z 0.0

# Create A300 (air node) at location (41.000005, 46.000005, 5.0) at time 10.
CreatePlatform A300 10.0 X 41.00005 Y 46.000005 Z 5.0 TYPE 1

# Command for node A100 to send 512-byte TCP messages to node A200 every
# 2 seconds, starting at time 150 seconds.
CommEffectsRequest  A100  A200 150.0  INTERVAL 2.0 ID1 1 ID2 0 TCP SIZE 512
DESCRIPTION "HELLO"

# Command for node A200 to send 512-byte TCP messages to node A100 every
# 2 seconds, starting at time 150 seconds.
CommEffectsRequest A200 A100 150.0 INTERVAL 2.0 ID1 1 TCP SIZE 512 DESCRIPTION
"HI"

# Command for node A300 to change its position at time 200 seconds.
# This command will result in an error because the coordinates are out
# of range.
UpdatePlatform A300 200.0 X 100.0 Y 400.0 Z 0.0

# Command to damage node A200 at 250.0 seconds.
UpdatePlatform A200 250.0 DamageState 1

# Command for node A100 to send 512-byte TCP messages to node A200 every
# 2 seconds, starting at time 350 seconds.
# This command will result in a failure because node A200 is damaged.
CommEffectsRequest A100 A200 350.0 INTERVAL 2.0 ID 1 TCP SIZE 512 DESCRIPTION
"BYE"
# Command to end simulation at time 600.
StopSimulation 600.0
```

FIGURE E-1. MTS Configuration File for Sample Scenario

E.3.2.2 QualNet Configuration File

To run MTS with QualNet, in addition to the parameters needed for stand-alone QualNet operation, several other parameters need to be specified in the QualNet configuration file. For the sample scenario described in Section E.3.1, the additional parameters that need to be specified are shown in Figure E-2. Comments before each parameter describe the purpose of the parameter. It is assumed that the parameter COORDINATE-SYSTEM is set to LANLOTALT in the QualNet configuration file. This is needed because the coordinates specified in the MTS configuration file in Section E.3.2.1 are in the Lat/Lon/Alt format.

```
# Indicate that Socket Interface will be used to communicate with another
# program.
SOCKET-INTERFACE YES

# Wait 15 seconds before sending a CommEffectsResponse FAILURE for
# a UDP message.
SOCKET-INTERFACE-UDP-FAILURE-TIMEOUT 15S

# Wait 90 seconds before sending a CommEffectsResponse FAILURE for a TCP
# message.
SOCKET-INTERFACE-TCP-FAILURE-TIMEOUT 90S

# Print per-packet statistics for each CommEffectsRequest to specified file.
SOCKET-INTERFACE-PRINT-PER-PACKET-STATS ./mts-demo.stats

# Specify the name of the entity mapping file.
SOCKET-INTERFACE-ENTITY-MAPPING-FILE ./mts-demo.entities

# Specify the number of ports Socket Interface should open for connection.
SOCKET-INTERFACE-NUM-PORTS 2

# Specify the ports to listen for the external program connection.
SOCKET-INTERFACE-PORT[0] 5032
SOCKET-INTERFACE-PORT[1] 5033

# Log all output to files.
SOCKET-INTERFACE-LOG FILE

# Base output statistics on simulation time (specify '0' as value).
SOCKET-INTERFACE-STATS-PRINT-REAL-TIME 0

# Set the interval to output statistics to 20 seconds.
SOCKET-INTERFACE-STATS-PRINT-INTERVAL 20

# Base the graph log file on simulation time (specify '0' as value).
SOCKET-INTERFACE-GRAPH-PRINT-REAL-TIME 0

# Set the interval to output the graph log file to 60 seconds.
SOCKET-INTERFACE-GRAPH-PRINT-INTERVAL 20S
```

FIGURE E-2.   Parameters to be Added to QualNet Configuration File for Sample Scenario

E.3.2.3 Entity Mapping File

For the sample scenario described in Section E.3.1, the entity mapping file is shown in Figure E-3. The entity mapping file is called mts-demo.entities. The left column corresponds to the MTS entity and the right column corresponds to the QualNet node identifier that the MTS entity maps to.

```
# Entity ID   QualNet Node ID

A100               5
"A200"            10
"A300"            "16"
```

FIGURE E-3.   Entity Mapping File for Sample Scenario

E.3.3 Output Files

In this section, we examine the output files produced by running the scenario described in Section E.3.1.

E.3.3.1 File driver.log

Figure E-4 shows the file driver.log corresponding to the sample scenario described in Section E.3.1. See Section 3.4.1 for a description of the syntax of the file driver.log.

- Each line starts with a timestamp (simulation time enclosed in square brackets).
- The first three lines correspond to simulation initialization.
- The next line corresponds to QualNet running in time managed fast mode with a time advance to 5.0 seconds.
- The next three lines correspond to the creation of the three nodes.
- The next several lines correspond to simulation time advancing by 5.0 seconds each time.
- Following that, the next line corresponds to the first (`Id2` value is 0) request to send data of size 512 bytes from node A100 to node A200. This corresponds to the following line from mts-config.txt (see Figure E-1):

```
CommEffectsRequest  A100  A200 150.0  INTERVAL 2.0 ID1 1 ID2 0 TCP SIZE
512  DESCRIPTION "HELLO"
```

- Most of the remaining lines correspond to exchange of data between node 100 and node 200 and to the advancing of the simulation clock. The value of `Id2` is incremented for each data unit exchanged between nodes.
- Figure E-4 also shows lines corresponding to update the position of node A300 and the command to stop the simulation.

```
[  0.000000] InitializeSimulation: TimeManagementMode = TimeManaged,
CoordinateSystem = LatLonAlt
[  0.000000] PauseSimulation
[  0.000000] ExecuteSimulation
[  0.000000] AdvanceTime: TimeAllowance = 5.000000
[  5.000000] CreatePlatform: EntityId = A300, Position = (41.000050, 46.000005,
5.000000), State = Undamaged, CreateTime = 10.000000, Type = 1
[  5.000000] CreatePlatform: EntityId = A100, Position = (41.000050, 46.000005,
0.000000), State = Undamaged, CreateTime = 10.000000, Type = 0
[  5.000000] CreatePlatform: EntityId = A200, Position = (41.000050, 46.000005,
0.000000), State = Undamaged, CreateTime = 10.000000
[  5.000000] AdvanceTime: TimeAllowance = 10.000000
[ 10.000000] AdvanceTime: TimeAllowance = 15.000000
[ 15.000000] AdvanceTime: TimeAllowance = 20.000000
...
[140.000000] AdvanceTime: TimeAllowance = 145.000000
[140.000435] CommEffectsRequest: Id1 = 1, Id2 = 0, Protocol = TCP, Size = 512,
Sender = A200, Receiver = A100, RequestTime = 150.000000, Description = HI
[140.000435] CommEffectsRequest: Id1 = 1, Id2 = 0, Protocol = TCP, Size = 512,
Sender = A100, Receiver = A200, RequestTime = 150.000000, Description = HELLO
[145.000000] AdvanceTime: TimeAllowance = 150.000000
[145.004444] CommEffectsRequest: Id1 = 1, Id2 = 1, Protocol = TCP, Size = 512,
Sender = A200, Receiver = A100, RequestTime = 152.000000, Description = HI
[145.004708] CommEffectsRequest: Id1 = 1, Id2 = 1, Protocol = TCP, Size = 512,
Sender = A100, Receiver = A200, RequestTime = 152.000000, Description = HELLO
[145.004708] CommEffectsRequest: Id1 = 1, Id2 = 2, Protocol = TCP, Size = 512,
Sender = A200, Receiver = A100, RequestTime = 154.000000, Description = HI
[145.007252] CommEffectsRequest: Id1 = 1, Id2 = 2, Protocol = TCP, Size = 512,
Sender = A100, Receiver = A200, RequestTime = 154.000000, Description = HELLO
[150.000000] AdvanceTime: TimeAllowance = 155.000000
...
[195.000000] UpdatePlatform: EntityId = A300, UpdateTime = 200.000000, Position
= (100.000000, 400.000000, 0.000000)
[195.000000] CommEffectsRequest: Id1 = 1, Id2 = 25, Protocol = TCP, Size = 512,
Sender = A200, Receiver = A100, RequestTime = 200.000000, Description = HI
[195.000000] CommEffectsRequest: Id1 = 1, Id2 = 25, Protocol = TCP, Size = 512,
Sender = A100, Receiver = A200, RequestTime = 200.000000, Description = HELLO
[195.000000] AdvanceTime: TimeAllowance = 200.000000
...
[595.000000] StopSimulation: StopTime = 600.000000
[595.000000] CommEffectsRequest: Id1 = 1, Id2 = 225, Protocol = TCP, Size =
512, Sender = A200, Receiver = A100, RequestTime = 600.000000, Description = HI
[595.000000] AdvanceTime: TimeAllowance = 600.000000
...
```

FIGURE E-4.   File driver.log for Sample Scenario

E.3.3.2 File errors.log

Figure E-5 shows the file errors.log corresponding to the sample scenario described in Section E.3.1. See Section 3.4.2 for a description of the syntax of the file errors.log.

In this scenario, the file errors.log records only one error which corresponds to the command to update Node A300's position to out-of-range coordinates.

```
[  0.000000] Socket Interface Errors Logfile
[200.000000] ErrorMessage: InvalidCoordinates, Error = Latitude out of range:
100.000000 (maximum is 42.000000), OriginatingMessage = UpdatePlatform:
EntityId = A300, UpdateTime = 200.000000, Position = (100.000000, 400.000000,
0.000000)
```

FIGURE E-5.   File errors.log for Sample Scenario

E.3.3.3 File graph.log

Figure E-6 shows the file graph.log corresponding to the sample scenario described in Section E.3.1. See Section 3.4.3 for a description of the syntax of the file graph.log.

Figure E-6 shows the connectivity graph at time 280.010899. It shows the status of each node. All three nodes are gateways of subnet 100. It also shows that node A300 is a RAP and there are no other nodes in its region.

```
...
[280.010899] ==============================================================
[280.010899] EntityID: A100              NodeID: 5
[280.010899]     Location: X = 41.000050, Y = 46.000005, Z = 145.426000
[280.010899]     Type: Ground
[280.010899]     Interface: 192.0.0.5      Subnet Id: 100
[280.010899]     Is Gateway:  Yes, subnets 100
[280.010899]     Multicast Groups: 224.0.0.5 224.0.0.1 224.0.0.2 224.0.0.13
[280.010899] EntityID: A200              NodeID: 10
[280.010899]     Location: X = 41.000050, Y = 46.000005, Z = 145.426000
[280.010899]     Type: Ground
[280.010899]     Interface: 192.0.0.10     Subnet Id: 100
[280.010899]     Is Gateway:  Yes, subnets 100
[280.010899]     Multicast Groups: 224.0.0.5 224.0.0.1 224.0.0.2 224.0.0.13
[280.010899] EntityID: A300              NodeID: 16
[280.010899]     Location: X = 41.000050, Y = 46.000005, Z = 5.000000
[280.010899]     Type: Air
[280.010899]     Interface: 192.0.0.16     Subnet Id: 100 [RAP]    Members:
[280.010899]     Is Gateway:  Yes, subnets 100
[280.010899]     Multicast Groups: 224.0.0.5 224.0.0.1 224.0.0.2 224.0.0.13
[280.010899] Region Summary.  Platforms in parenthesis are RAPs.
[280.010899]     Subnet 100, gateways A100 A200 A300
[280.010899]         (A300)
[300.012106] ==============================================================
...
```

FIGURE E-6.   File graph.log for Sample Scenario

E.3.3.4 File responses.log

Figure E-7 shows the file responses.log corresponding to the sample scenario described in Section E.3.1. See Section 3.4.4 for a description of the syntax of the file responses.log.

- Each line starts with a timestamp (simulation time enclosed in square brackets).
- The first few lines correspond to simulation initialization.
- The next several lines correspond to responses to command to advance time.
- The next several lines, the first one with timestamp 151.310207, correspond to successful responses communication effects requests.
- At time 256.000435, a response indicating a failure is sent. Note that this response corresponds to the request which was received 90.000435 seconds earlier. This latency is due to the TCP timeout being set to 90 seconds in the QualNet configuration file.

```
...
[  0.000000] Socket Interface Respones Logfile
[  0.000000] SimulationState: State = Standby, OldState = Initialized
[  0.000000] SimulationState: State = Initialized, OldState = Paused
[  0.000000] SimulationState: State = Paused, OldState = Executing
[  4.998801] SimulationIdle: CurrentTime = 4.998801
[  9.999403] SimulationIdle: CurrentTime = 9.999403
[ 14.998114] SimulationIdle: CurrentTime = 14.998114
...
[149.999423] SimulationIdle: CurrentTime = 149.999423
[151.310207] CommEffectsResponse: Id1 = 1, Id2 = 0, Sender = A200, Receiver =
A100, Status = Success, ReceiveTime = 151.310207, Latency = 1.310207
[151.728266] CommEffectsResponse: Id1 = 1, Id2 = 0, Sender = A100, Receiver =
A200, Status = Success, ReceiveTime = 151.728266, Latency = 1.728266
[152.630395] CommEffectsResponse: Id1 = 1, Id2 = 1, Sender = A100, Receiver =
A200, Status = Success, ReceiveTime = 152.630395, Latency = 0.630395
[152.660400] CommEffectsResponse: Id1 = 1, Id2 = 1, Sender = A200, Receiver =
A100, Status = Success, ReceiveTime = 152.660400, Latency = 0.660400
...
[164.310064] CommEffectsResponse: Id1 = 1, Id2 = 7, Sender = A200, Receiver =
A100, Status = Success, ReceiveTime = 164.310064, Latency = 0.310064
[164.999260] SimulationIdle: CurrentTime = 164.999260
...
[254.999954] SimulationIdle: CurrentTime = 254.999954
[256.000435] CommEffectsResponse: Id1 = 1, Id2 = 8, Sender = A200, Receiver =
A100, Status = Failure, ReceiveTime = 256.000435, Latency = 90.000435,
Description = HI
[256.000435] CommEffectsResponse: Id1 = 1, Id2 = 8, Sender = A100, Receiver =
A200, Status = Failure, ReceiveTime = 256.000435, Latency = 90.000435,
Description = HELLO
...
```

FIGURE E-7.   File responses.log for Sample Scenario

E.3.3.5 File stats.log

Figure E-8 shows the file stats.log corresponding to the sample scenario described in Section E.3.1. See Section 3.4.5 for a description of the syntax of the file stats.log.

- The first line states that the statistics printed on the following lines are for the time interval from the start of simulation to simulation time 200.000070 seconds.

- The second line shows the simulation time and real time elapsed.

- The next line shows how many platforms have been created (number of `CreatePlatform` messages processed), how many platforms have been updated (number of `UpdatePlatformState` messages processed), how many communication requests have been modeled (number of `CommEffectsRequest` messages processed), and the total number of `CreatePlatform`, `UpdatePlatformState`, and `CommEffectsRequest` messages processed.

- The next line shows how many communication responses (`CommEffectsResponse` messages) were generated as well as how many of those were successful and how many were failures. (Note that some communication requests may be outstanding.)

- The next line shows the average number of messages that were received since the beginning of the simulation (`Avg Mesg/Sec`) as well as the average number `CommEffectsResponse` messages sent as successes (`Succ`) and failures (`Fail`).

  The last item printed on the same line is the average number of late packets, i.e., `CommEffectsResponse` messages that were designated as failures but would have been successes if the failure timeout window were larger. For example, if a packet takes 20 seconds to reach its destination but the failure timeout window is 15 seconds, then the packet is marked as a failure even though it was received successfully.

- The next line shows the average number of messages (`Last Mesg/Sec`), the average number `CommEffectsResponse` messages sent as successes (`Succ`) and failures (`Fail`), and the number of late packets in the period from the time of the last statistics log to the current simulation time.

- The final line shows the average real time speed since the beginning of simulation (`Avg Real Time Speed`) and average real time speed in the period from the time of the last statistics log to the current simulation time (`Last Real Time Speed`). This indicates the speedup of QualNet versus real time. A real time speed of 10 means QualNet is running 10 times faster than real time.

```
...
[200.000070] Performance Stats for Time Interval: SimTime = 200.000070
[200.000070] Current Sim Time[s] =          200.00   Real Time[s] =     2.85
[200.000070]     Creates = 3, Updates = 0, Requests = 62, Messages = 65
[200.000070]      Responses = 16, Successes = 16, Failures = 0
[200.000070]      Avg  Mesg/Sec = 22.84, Succ = 5.62, Fail = 0.00 (0.00 late)
[200.000070]      Last Mesg/sec = 77.01, Succ = 0.00, Fail = 0.00 (0.00 late)
[200.000070]     Avg  Real Time Speed = 43.35, Last Real Time Speed = No Packets
[220.000435] Performance Stats for Time Interval: SimTime = 220.000435
[220.000435] Current Sim Time[s] =          220.00   Real Time[s] =     3.12
[220.000435]     Creates = 3, Updates = 0, Requests = 82, Messages = 85
[220.000435]      Responses = 16, Successes = 16, Failures = 0
[220.000435]      Avg  Mesg/Sec = 27.23, Succ = 5.13, Fail = 0.00 (0.00 late)
[220.000435]      Last Mesg/sec = 76.21, Succ = 0.00, Fail = 0.00 (0.00 late)
[220.000435]     Avg  Real Time Speed = 43.35, last Real Time Speed = No Packets
...
```

FIGURE E-8.   File stats.log for Sample Scenario