



Event Router Getting Started Guide

© KnowNow, Inc.
100 View Street, Suite 101
Mountain View, CA 94041

+1.650.641.8500
<http://www.knownow.com>

Trademarks

KnowNow™. The KnowNow logo and icon are trademarks of KnowNow, Inc.

Other product names used herein may be trademarks or registered trademarks of KnowNow, Inc. or other companies.

Statement of Conditions

KNOWNOW, INC. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL KNOWNOW BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, EVEN IF KNOWNOW HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES ARISING FROM ANY DEFECT OR ERROR IN THIS PUBLICATION OR IN THE KNOWNOW SOFTWARE.

KnowNow, Inc. may revise this publication from time to time without notice. Some states or jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

Legal Notices

Copyright© 2000 - 2002 KnowNow, Inc. All rights reserved.

All rights reserved. No part of this work covered by copyright herein may be reproduced in any form or by any means - graphic, electronic or mechanical—including photocopying, recording, taping, or storage in an information retrieval system, without prior written permission of the copyright owner.

Contents

Preface

Intended Audience.....	vii
What This Guide Covers	vii
Before You Start.....	vii
Document Conventions	viii
How the Guide Is Organized	viii
Related Documentation.....	viii
How to Contact KnowNow Inc.	x

Chapter 1 - Overview

Event Router Concepts	1-2
Event Router Terminology	1-2
Events	1-3
Topics	1-4
Routes	1-4
Metadata Subtopics	1-5
Subscriptions vs. Routes, and Journal Topics	1-6
Event Lifetimes and Initial Route Population	1-7
Using the Event Router	1-8
Basic Use	1-8
Aggregation	1-9

Chapter 2 - Installation and Configuration

Installing the KnowNow Event Router	2-2
Upgrading a Previous Install	2-2
Installing the Event Router as a Service on Windows 2000/XP	2-3
Installing the Event Router on Solaris or Linux	2-7
Starting the Router Automatically	2-8
Stopping the Router	2-8
On Solaris/Linux	2-8
On Windows	2-8
Verifying that the Router is Running	2-9
Configuring the KnowNow Event Router	2-10
Command-line Options	2-11
Windows 2000/XP Usage:	2-11

Solaris/Linux Usage: 2-11
 nsd Command Options for Windows/Solaris/Linux 2-11
Changing the Router Desktop Install to Install as a Service 2-12
 To install the service 2-12
 To start the service 2-12
 To ensure the service is running 2-13
 To stop the service 2-13
 To remove the service 2-13

Chapter 3 - Router Utilities

Router Utilities 3-1
Using the Utilities 3-2
KN Config Utility 3-3
 Starting the Config Tool 3-3
 The Config Tool Screen 3-4
 Config Editor 3-5
 Router Data 3-5
 Using the Config Editor 3-6
 Saving the Edited Config Data 3-8
 Restoring the Config Data 3-8
 Exiting the Config Tool 3-8

Chapter 4 - Managing the KnowNow Event Router

Access Control in the KnowNow Event Router 4-2
 passwd 4-2
 group 4-3
 perms 4-3
Authentication/Authorization modules 4-5
 Using Authentication and Authorization to Restrict Access 4-6
 Restricting Topics to Authenticated Users only 4-7
 Restricting the kn_config topic 4-8
 Restricting Access to Various Event Router Operations 4-8
 Deleting a topic 4-8
 Attaching filters 4-9
 Fetching and replacing the contents of the config file 4-9
 Setting default expiration time 4-9
 Setting maximum queue length 4-10
Clustering 4-11
 What is Clustering? 4-11
 Failover Mechanism 4-11
 Scalability of Connections 4-12
 How to Cluster? 4-13
 Important Clustering Considerations: 4-15
Persistence 4-16
 persistence_file 4-17

persistence_interval	4-17
persistence module	4-17
atomic_router	4-17
atomic_topic	4-18
Moving Persistence Information	4-18
SSL Configuration	4-19
Generating and Installing Certificates	4-20
Chapter 5 - Tuning Parameters	
KnowNow Event Router Tuning Parameters	5-2
Chapter 6 - Frequently Asked Questions	
Frequently Asked Questions	6-2

Preface

Intended Audience

The KnowNow *Event Router Getting Started Guide* is designed for developers who are installing and configuring the KnowNow Event Router.

What This Guide Covers

This guide describes how you can install the KnowNow Event Router, configuration, Router terminology and concepts, clustering and SSL.

Before You Start

This guide is written for programmers, so it is assumed that the reader possesses a general knowledge of event-driven programming and HTTP/HTML as well as specific knowledge of the C and C++ programming languages.

Document Conventions

<i>Italic</i>	Italics are used the first time a glossary term is introduced, for the titles of books, and for menu items and options.
<ul style="list-style-type: none">• Bulleted lists	Bulleted lists designate items of equal importance.
<ol style="list-style-type: none">1. Numbered lists	Numbered lists designate a specific sequence of steps required to complete a procedure.
Boldface type	Boldface type is used for emphasis.
Code	Code excerpts and command line sequences are shown in this type face.
Ellipsis . . .	Is used in code and argument syntax to indicate that inconsequential information is not shown.

How the Guide Is Organized

[Chapter 1, “Overview,”](#) provides an overview of the Event Router, including terminology and basic concepts.

[Chapter 2, “Installation and Configuration,”](#) describes how to install and configure the Event Router.

[Chapter 3, “Router Utilities,”](#) describes how to use the Router Utilities.

[Chapter 4, “Managing the KnowNow Event Router,”](#) describes how to use access control to restrict users and also clustering and SSL.

[Chapter 5, “Tuning Parameters,”](#) provides a list of KnowNow Event Router parameters and their descriptions.

[Chapter 6, “Frequently Asked Questions,”](#) describes frequently asked questions regarding the Event Router installation and configuration.

[Glossary,](#) provides a list of KnowNow terms and their descriptions.

Related Documentation

KnowNow Microserver for Java Programmer’s Guide

This guide describes how you can use the Microserver for Java application programming interfaces (APIs). The API reference describes all of the available methods, functions, and formats.

KnowNow Event Router Programmer's Guide

This guide describes how you can use the features of the KnowNow Event Router application programming interface (APIs) and the Microserver for JavaScript.

The API reference section describes all of the HTTP-based and JavaScript interfaces.

KnowNow C Microserver for Solaris Guide

This guide describes how you can use the features of the Microserver C application programming interfaces (APIs). The API reference describes all of the available functions and data types.

Microserver for Windows Programmer's Guide

This guide describes how you can use the features of the Microserver Windows application programming interfaces (APIs). The API reference describes all of the available methods, functions, and formats.

How to Contact KnowNow Inc.

We welcome any comments you may have about our software products or our documentation. Our goal is to provide functional and easy-to-use products that help you to work more efficiently. Please mail or fax your comments to the following:

KnowNow Inc.
100 View Street
Suite 101
Mountain View, CA 94041
Fax: 650-641-8501
email: <http://developer.knownow.com>

Overview

The KnowNow Event Router manages topics and the flow of events from publishers to subscribers. This chapter discusses basic Router concepts and basic usage scenarios for the KnowNow Event Router. The following sections are discussed:

- [Event Router Concepts](#)
- [Basic Use](#)

Event Router Concepts

The KnowNow Event Router accepts events as input, then *routes* them to other destinations. This is very comparable to an IP router: it accepts packets from one network and sends them to a specified destination. An IP router interconnects networks; a KnowNow Event Router interconnects applications. Applications *publish* events to the Router, and the Router copies those events to other applications which have *subscribed* to them. Currently, the KnowNow Event Router accepts published events via HTTP as well as subscription requests via HTTP; other protocols are possible, but have not yet been implemented.

Event Router Terminology

Router Term	Description
Events	Messages sent by an application or person specifically to trigger actions.
Topics	A collection of events to which KnowNow clients can subscribe or publish events.
Routes	The path an event travels during the publish/subscribe process.
Publish	The act of sending an event from a KnowNow Event Router.
Subscribe	The act of receiving an event from a KnowNow Event Router.

Events

An *event* is a basic packet of information that the Router knows how to deal with. KnowNow events are made up of key/value pairs, for example, "price: %9.03". Applications can use any key/value pairs that make sense for their application; KnowNow reserves all headers beginning with `kn_` for its own use and definition.

A typical event might look like the following:

```
kn_id: 141059u9103
kn_time_t: 996172943
price: 9.03
qty: 1000
symbol: SUNW
```

This event contains three headers defined by the application (`price`, `qty`, and `symbol`) and two `kn_headers` that have special meaning:

- `kn_id` holds a unique identifier for this event. Applications can generate their own id's (by including a `kn_id` header and value when the event is published to the Router) or can rely on the Router to generate them. If an event is published to a Router without a `kn_id` field, the Router will automatically generate and apply a unique `kn_id` header.
- `kn_time_t` contains the time (measured in seconds since Jan. 1, 1970) this event was first published. The Router applies this header to any event it receives which does not already contain a `kn_time_t` field. In the example above, the event was first published on Thursday, July 26, 2001 at 11:42:23am.

`price`, `qty`, and `symbol` are application-defined headers which have meaning to the publisher. Note that the KnowNow Event Router does not place any restrictions on these headers; non-ASCII data need only be encoded in UTF-8 format before transmission into the Router.

Topics

Every time an event is published to the Router, it must be published to a topic. Think of a topic as the address to which an event gets published (topics are also important for subscriptions). Every topic on a KnowNow Event Router is fully addressable as a URI. For example, the `topic/stockQuotes` on the Router running on `gravitar.bigcompany.com` can be addressed as:

```
http://gravitar.bigcompany.com/kn/stockQuotes
```

A topic name can contain roman letters, digits, slashes, underscores, dashes, periods, and non-ASCII characters.

You can create a topic just by referencing the topic. Events are sent to a topic by a publisher and then the topic forwards that event along all of its *routes* to all the subscribers of that topic.

The KnowNow Event Router manages a hierarchy of topics much like a standard web server manages a hierarchy of URLs. For example, an application might use `/stockQuotes/nasdaq` to publish a feed of Nasdaq price updates, and `/stockQuotes/nyse` for publishing NYSE updates. Note that if an event is published to `/stockQuotes`, only subscribers to that topic will be sent a copy of the event; subscribers to `/stockQuotes/nyse` will not receive the event. The topic hierarchy is initially empty when you start up a brand new Router, and topics get created on the fly as events are published into them.

Routes

One of the most powerful things the KnowNow Event Router allows you to do is set up *routes* out of a topic. When an event is published into a topic, the Router automatically forwards that event along all the routes out of the topic. Routes can be made to any URI (currently only HTTP and HTTPS protocols are supported). This means, for instance, that you could create a route from the topic `/stockQuotes/nyse` to CGI script or servlet on another machine (e.g. a route to `http://weblogic.bigcompany.com/servlet/nyseListener`).

Whenever an event is published into `/stockQuotes/nyse`, the Router makes an HTTP POST request to the URI referenced in the route, sending the event a part of the request. Since the destination of a route can be any URI, routes can be made between topics on the same KnowNow Event Router, different Routers, or from the Router to any application that can receive an HTTP POST request. In addition, you can create an unlimited number of routes out of a given topic, making fan-out of events easy to do.

Metadata Subtopics

The Router maintains metadata subtopics for every topic in its hierarchy. These are called `kn_routes` subtopics and `kn_subtopics`. The following example illustrates what these topics do and how they can be used within a topic hierarchy:

```

/stockQuotes
  /stockQuotes/kn_subtopics
  /stockQuotes/kn_routes
  /stockQuotes/nyse
    /stockQuotes/nyse/kn_routes
    /stockQuotes/nyse/kn_subtopics
  /stockQuotes/nasdaq
    /stockQuotes/nasdaq/kn_subtopics
    /stockQuotes/nasdaq/kn_routes

```

`kn_subtopics` topics contain an event for each of the subtopics of the given topic. In the preceding example, `/stockQuotes/kn_subtopics` contains an event for `/stockQuotes/nyse`, one for `/stockQuotes/nasdaq`, and one for `/stockQuotes/kn_routes`. The Router will publish an event into the `kn_subtopics` topic whenever a subtopic gets created; thus if the subtopic `/stockQuotes/ftse` were created, a new event would be published to `/stockQuotes/kn_subtopics` referring to the new subtopic. Note that `kn_subtopics` topics can be subscribed to just like any other topic, making it possible for an application to find out when new topics are created as well as to enumerate the existing topic hierarchy. However, only the Router may publish directly to the `kn_subtopics` topic.

Similarly, `kn_routes` topics contain an event for each of the routes out of a given topic. For instance, if a route were created from `/stockQuotes/nasdaq` to a new topic called `/stockQuotes/all`, the Router would publish an event into `/stockQuotes/nasdaq/kn_routes` with the full URI of the destination topic (in this case, `http://router.bigcompany.com/kn/stockQuotes/all`). Just as with `kn_statistics`, it is possible to subscribe to `kn_routes` topics; applications can then receive notification whenever a new route is created out of a given topic. Only the Router may publish to a `kn_routes` topic.

Subscriptions vs. Routes, and Journal Topics

The KnowNow Event Router does not distinguish between a subscriber to a topic and a route out of a that topic. As the Router only routes to URIs, what is referred to as a *subscription* in another Publish and Subscribe system is just a *route* out of a *topic* in the KnowNow Event Router. At the very simplest conceptual level, what the Router does when it receives an event to publish is as follows:

- Determines the topic to which the event is destined
- Creates that topic if it does not already exist (and adds a `kn_subtopics` entry to the new topic's parent topic)
- Delivers the event into the topic
- For each entry in the topic's `kn_routes` subtopic, sends the event to the URI specified in the route. Current Event Routers only support HTTP and HTTPS protocol URIs; the Router sends the event data as an HTTP POST to the specified location.

The KnowNow Microservers allow applications to maintain a persistent connection to the Router. Therefore, when an application subscribes (creates a route out of) a particular topic, any events published to that topic are sent down to the application over a live TCP connection instead of the Router making an HTTP POST to the application. Microservers accomplish this using a third type of special topic known as a *journal* topic.

A journal topic is any topic whose final path element is `kn_journal`. For example: `/who/jenkins/kn_journal` would be a valid journal topic; while `/who/jenkinskn_journal` would not be.

Journal topics are special in that:

- they do not have subtopics (no `kn_subtopics`)
- they cannot have routes out of them (no `kn_routes` subtopic).

Journal topics can only be used as the target of a route from another topic (e.g. a route from `/stockQuotes/nasdaq` to `/who/steve/kn_journal`). Journal topics are the endpoints to which Microservers can connect to create *tunnels*, the persistent TCP connections to the Router.

When an application initializes a Microserver, the Microserver connects to the Router (using an HTTP GET request) and creates a tunnel from a random journal topic name. Some Microservers use a naming convention for their journal topics such that they all begin with `/who`, but this is not required by the Router. As long as the journal topic ends with `/kn_journal`, the Router will create the journal and associated tunnel.

Later, when the application asks the Microserver to subscribe to a given topic (e.g. subscribe to `/stockQuotes/nyse`), the Microserver actually creates a route out of `/stockQuotes/nyse` to the journal topic associated with its connection. When an event is later published to `/stockQuotes/nyse`, it is forwarded along the route to the user's journal, and from there sent down the tunnel to the Microserver.

You can have more than one tunnel out of a single journal, which makes it possible for more than one Microserver to connect to the same stream of events. In the case where a single journal topic has more than one tunnel associated with it, any event delivered to the journal topic will be sent down all of the connected tunnels.

The Router currently supports two kinds of tunnels: *simple tunnels* and *javascript tunnels*. Microservers select the kind of tunnel to create when they connect to the Router. Currently, all Microservers use simple tunnel format to communicate with the Router, with the exception of the JavaScript Microserver.

Finally, journal topics support one extra feature, *checkpointing*. All events delivered to a journal topic are modified to include a `kn_route_checkpoint` header. The value of this header is a unique identifier for the event that is used when reconnecting to a pre-existing journal. In the case where a Microserver's tunnel connection is broken for some reason, it can reconnect to its journal topic and provide the last checkpoint ID if successfully received. The Router then sends all events since the checkpoint down the tunnel, making it possible for the application to be sure it has received all its events, despite the reconnect.

Event Lifetimes and Initial Route Population

When an event is delivered to a topic, it is stored inside the topic and then forwarded along any routes out of the topic. Unlike other publish and subscribe systems, the Router allows events to live beyond the time of their initial publishing. Events can include a `kn_expires` header that sets the time the event will expire. Events published to the Router that do not include this header are subject to a default expiration time, configurable in the Router's runtime configuration. (The shipping default of this parameter is currently 3600 seconds, or one hour.)

When a route is created out of a topic, the route creation request can include parameters for initial route population (IRP), which requests the Router to immediately forward existing non-expired events along the route. Route requests can ask the Router to deliver the last n events (known as `do_max_n`) or all the events seen within the previous n seconds (`do_max_age`).

Using the Event Router

This section describes a couple of KnowNow Event Router usage scenarios:

- Basic Use
- Aggregation

Basic Use

The most basic use of the Router is as follows:

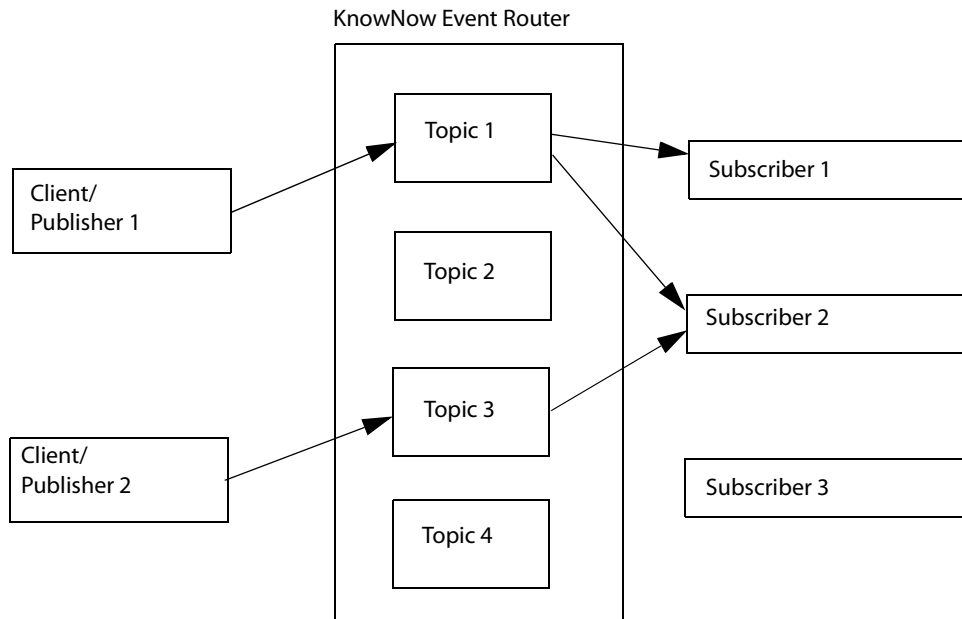
- A client publishes events to a topic, which resides on a KnowNow Event Router.
- The topic then immediately passes the events to all the subscribers that are subscribed to that topic.

Subscribers can also subscribe to multiple topics.

In the following example:

- Two subscribers (Subscriber 1 and 2) have subscribed to Topic 1.
- Subscriber 2 has also subscribed to Topic 3.
- When the client/publisher publishes an event to Topic 1, both subscribers will be notified.
- If the client/publisher were to publish to Topic 3, only Subscriber 2 would be notified.

Figure 1-1. Using the KnowNow Event Router - Basic Scenario



Aggregation

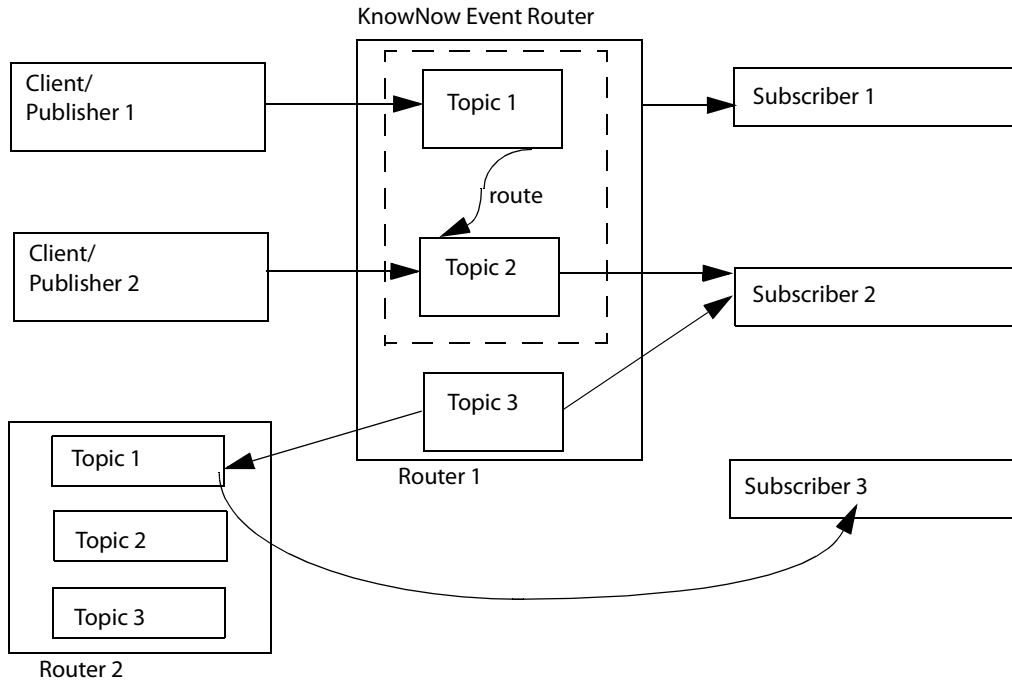
Topics within a KnowNow Event Router can also be subscribed to one another. In this way, events transmitted on a set of topics may feed into one master topic. In addition, events can be transmitted to topics on another KnowNow Event Router via aggregation.

In the scenario below:

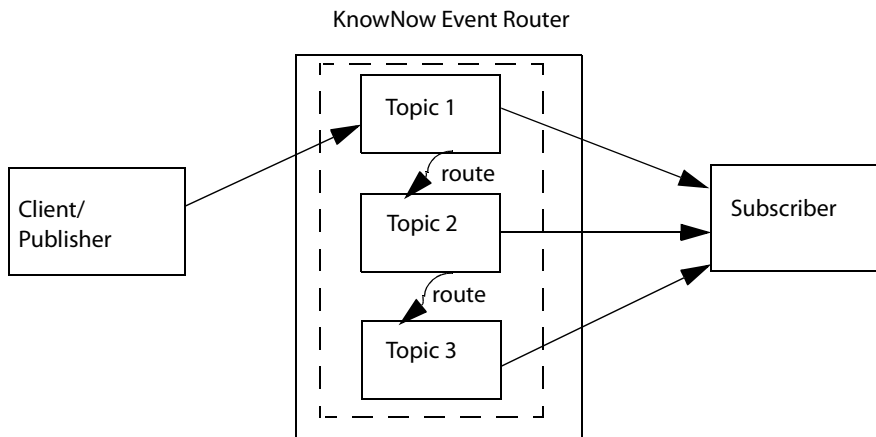
- Events published to Topic 1 are delivered to Subscriber 1, as well as Topic 2. Topic 2 then delivers the events to its subscribers (Subscriber 2).
- Events published directly to Topic 2 are sent only to Subscriber 2.
- Events published to Topic 3 are delivered to Subscriber 2 as well as to Topic 1 on the other Router. Topic 1 on the other Router then delivers the events to Subscriber 3. See [Figure 1-2](#).

Topics and Routes are managed by applications which are written to work with the KnowNow Router through the use of KnowNow Microservers. More information about these Microservers can be found at Developer.KnowNow.com.

Figure 1-2. Using the KnowNow Event Router - Aggregation Scenario



Special case—subscribing to two topics along a route (see figure below):



If you subscribe to two or more topics along a route, you get the same event more than once.

Installation and Configuration

This chapter describes installation and configuration procedures for the KnowNow Event Router. The following topics are discussed:

- [Installing the KnowNow Event Router](#)
- [Starting the Router Automatically](#)
- [Stopping the Router](#)
- [Verifying that the Router is Running](#)
- [Configuring the KnowNow Event Router](#)
- [Changing the Router Desktop Install to Install as a Service](#)

Installing the KnowNow Event Router

This section takes you through installing the KnowNow Event Router on Windows 2000/XP, Solaris, and Linux. After you install the Router, verify that the Router is working. For more information, see [“Verifying that the Router is Running” on page 2-9](#).

Upgrading a Previous Install

If you have a previous installation of the KnowNow Event Router, please make sure you do the following before you upgrade to a newer version:

- Run `db_kncopy.exe` on the `runtime/knrouter.prs` file to copy the persistence database to the new installation location (i.e., if you want to save your persistence data).
- For Linux and Solaris installs, copy your configuration file (`knrouter.conf`) to save your custom configuration settings. This way, if for any reason, you need to uninstall the Router, your previous settings can be restored. The `knrouter.conf` file is located in:

`event_router/conf/knrouter.conf`

For Windows, the `conf` directory is automatically backed up from the previous install (during upgrade), so that you will not lose any of your configuration, perms, or group settings.

- For version 1.7, you will need to install a new license key, even if the old license key is not yet expired.

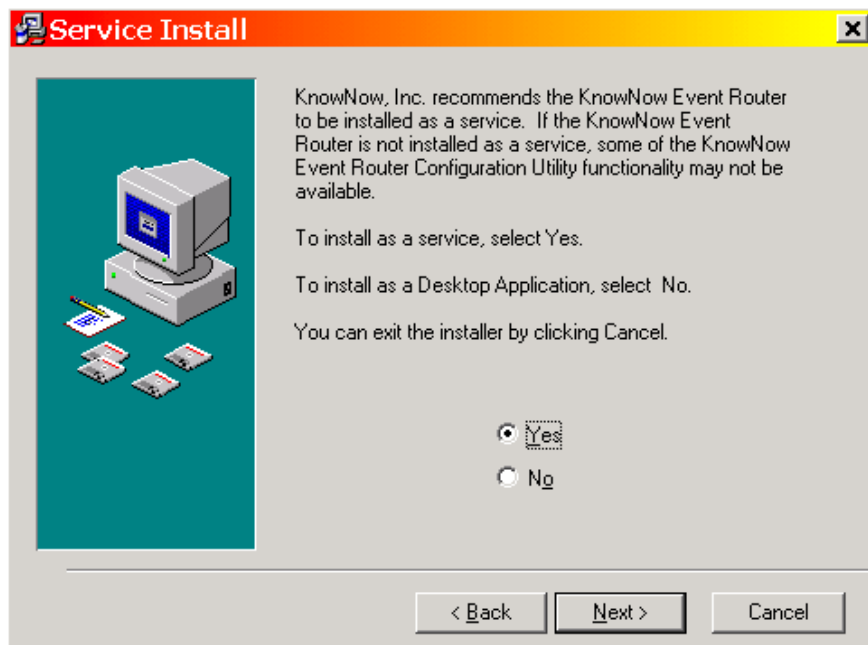
Installing the Event Router as a Service on Windows 2000/XP

Important: Before you can install the Event Router, please ensure that you have Administrator or user with administrative privileges. Otherwise, the Event Router will be installed as a desktop application.

To install the KnowNow Event Router on Windows 2000/XP:

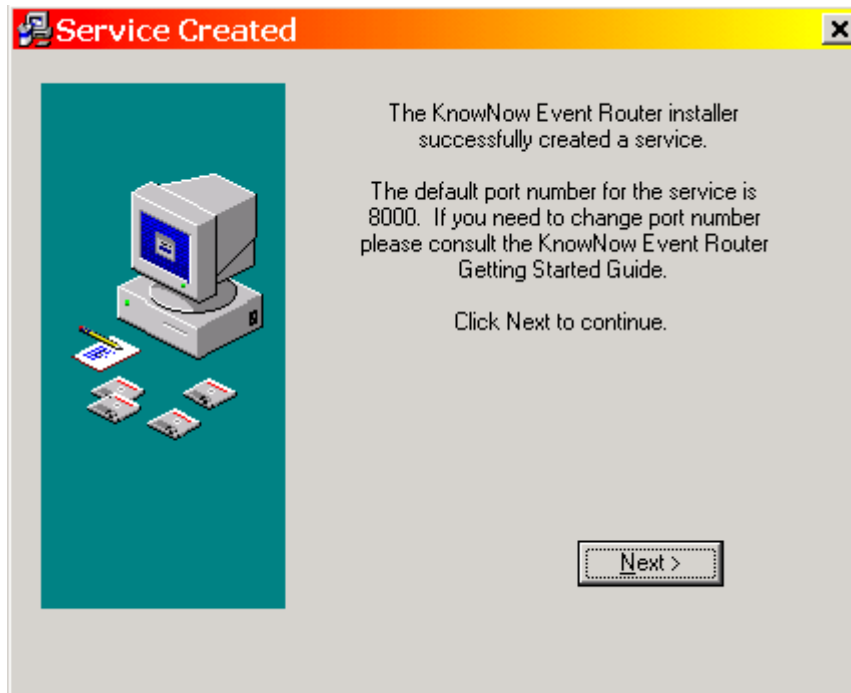
1. Run the Event Router installer executable. After the initial Welcome Message, the following screen displays asking you if you want to install the Event Router as a service.

Figure 2-1. Install as a Service



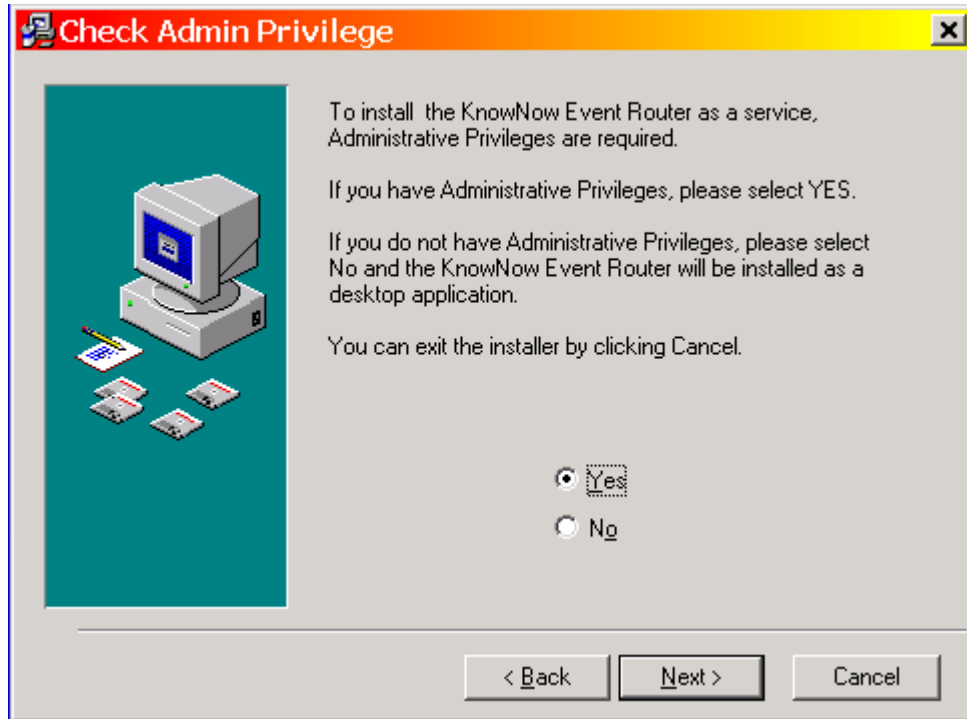
2. If you have the proper administrative privileges to install the Event Router as a service and select Yes, the Event Router is installed as a service and the following message displays informing you that the installation was successful.

Figure 2-2. Service Created



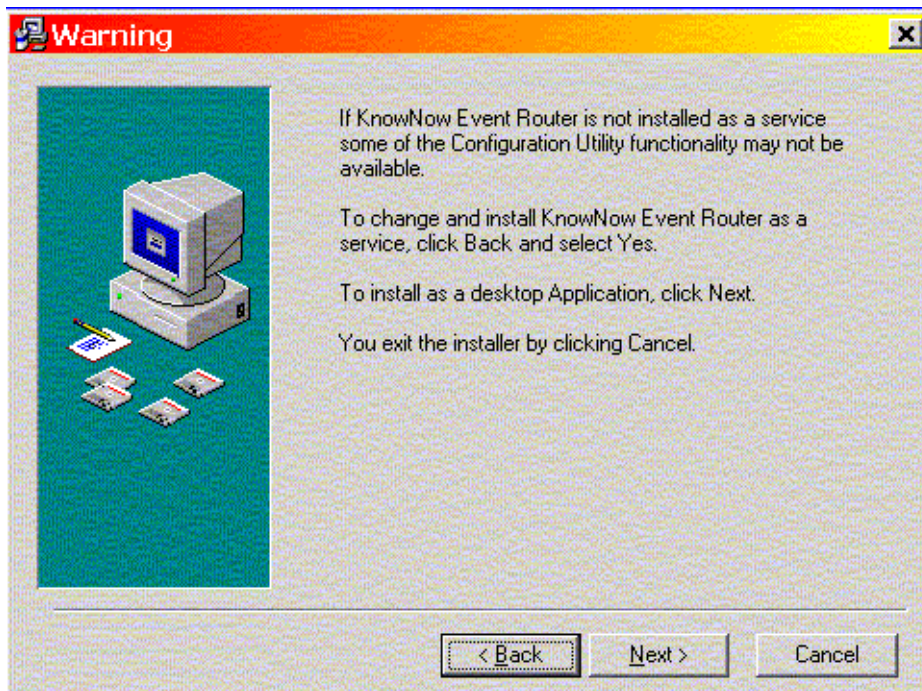
3. However, if you do not have administrative privileges, a warning panel is displayed and the Event Router will be installed as a desktop application.

Figure 2-3. Admin Privileges



4. If you do not want to install the Event Router as a service, select No on the Install as a Service screen. The following warning displays informing you that some of the Event Router Configuration Utility functions will not work if the Event Router is not installed as a service. You may still choose to install the Event Router as a desktop application and later change the installation to a service. For more information on that, refer to [Changing the Router Desktop Install to Install as a Service](#).

Figure 2-4. Warning



5. Once the Event Router has been installed, place the **knrouter.lic** license key file in the **\Program Files\KnowNow\Event_Router\conf** subdirectory. The **knrouter.lic** file is sent to you via email when you purchase the Router or request an evaluation copy of the product. If you have not received a license file, please contact support@knownow.com. If you already have the **knrouter.lic** file, the installer will prompt you for the license file and copy the file in the `knrouter.conf` subdirectory.
6. If the Event Router was not installed as a service and was installed as a Desktop Application, click the *Start Event Router* icon, which the installer created, to start the Router.

Installing the Event Router on Solaris or Linux

Before you install the KnowNow Event Router and if you are running Solaris 2.8 or an earlier version, you must download the `/dev/random` patch for Solaris 2.8. Solaris 2.9 already has this patch built-in, so this step is not required.

The README file and the patch can be found at:

- <http://access1.sun.com/patch.public/patches/all/patch.readme/README.112438-01>
- http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fpatches%2F112438&zone_32=112438



Note: Make sure you REBOOT after applying the patch.

To install the KnowNow Event Router on Solaris/Linux:

1. Uncompress and extract the tar file within a directory of your choice. This creates a **/usr/local/kn** subdirectory in the directory where the tar file was extracted. Optionally, you can login as root and copy the files into **/usr/local/kn**.
2. Place the **knrouter.lic** license key file in the **/usr/local/kn/conf** subdirectory. The **knrouter.lic** file is sent to you via email when you purchase the Router or request an evaluation copy of the product. If you have not received a license file, please contact support@knownow.com
3. Start the Router:
 - a. Change to the **/usr/local/kn** directory. This is the installation directory on solaris/linux. When you change to it, you are making it your current local directory.
 - b. Start the Router in the foreground or the background. See commands below:
starting in foreground: `bin/nsd -f -t conf/knrouter.conf`
starting in background: `bin/nsd -t conf/knrouter.conf`



Note: By default, the Router listens for requests on port 8000. To change the port number, edit the **httpport** variable at the top of the config file. You must restart the Router for the changes in the port number to take effect.

Starting the Router Automatically

To start the Router automatically when the machine reboots, the Solaris/Linux administrator needs to create a script in `/etc/init.d` for the Router and then symlink into that script form `/etc/rc2.d` or `/etc/rc3.d`.

Stopping the Router



Note: You can shut down the Router using the `do_method=shutdown` command only if you have the permission to modify the `/kn_config` topic.

On Solaris/Linux

If for some reason you want to stop the Router, you can use the following command:

```
pkill nsd
```

Or,

```
kill pid (process id for the nsd)
```



Note: If you are using the `persistence_module`, do not call `kill-9` to stop the Router. `kill-9` will not give the Router a chance to save to a database.

On Windows

If you are running the Event Router as a service and want to stop the service, do the following:

1. From the *Start* menu, select *Settings/Administrative Tools/ Services*.
2. Select the KnowNow Event Router.
3. Right-click the mouse button and select *Stop* service.

If you are running the Event Router as a desktop application, do the following:

1. For foreground mode, in the command line, type `Ctrl+C`.
2. For background mode, use the Task Manager. Select the KnowNow Event Router executable (`nsd.exe`) and select *End Task*.

Verifying that the Router is Running

To verify that the Router you just installed is running, do the following:

1. Open a web browser and connect to **http://hostname:8000/**

where hostname is the name or IP address of the machine running the KnowNow Event Router. The Router binds to port 8000 by default, this can be changed via the **httpport** setting in the Router's configuration file.

2. If you are not able to connect to the Router, you can check the log file written to the **log** subdirectory (`/usr/local/kn/log`) which is created when the Router is run as a daemon (or as a service on Windows). The `server.log` file contains diagnostics from the Router itself. There's a separate `access.log` file which contains an entry for every HTTP connection to the Router.

You must also verify that any firewalls between the client and server are set up to allow connection on the chosen port.

For example, If your Router is running on a machine named *router10 port 9500*, you would connect to **http://router10:9500/**. This command brings up a page with links to several KnowNow sample applications as well as documentation files.

Configuring the KnowNow Event Router

After you have installed the Router, you may want to change a few Router configuration parameters. This is done by editing the **knrouter.conf** file, which is located in the **conf** subdirectory. To help you through a simple configuration process, a sample **knrouter.conf** file has been provided in the KnowNow Event Router software package. The sample file has the following settings:

- The **httpport** parameter = 8000, so the Router listens for requests on port 8000.
- The **persistence_interval** parameter = 600 (seconds), so the Router persists its state to disk every 10 minutes.
- No permissions are configured in the **perms** file, so no credentials are required to access any topic on the Router.
- Clustering is disabled by default.

For more information on the Router parameters listed above, see [Chapter 5, "Tuning Parameters"](#) and for more information on the perms file, see [Chapter 4, "Managing the KnowNow Event Router"](#).

Command-line Options

The following command-line options are available for KnowNow Event Router configuration.

Windows 2000/XP Usage:

```
nsd [-h|V] [-i|f] [-z] [-q] [-I\R] [-s <server>] -t<file>
```

Solaris/Linux Usage:

```
nsd [-h|V] [-i|f] [-z] [-q] [-d] [-u <user>] [-g <group>] [-r <path>]
[-s <server>] -t <file>
```

nsd Command Options for Windows/Solaris/Linux

```
-h help (this message)
-V version and release information
-i initab mode
-f foreground mode
-z zippy memory allocator
-q non-quiet startup
-d debugger-friendly mode (ignore SIGINT) [solaris/linux only]
-u run as <user> [solaris/linux only]
-g run as <group> [solaris/linux only]
-r chroot to <path> [solaris/linux only]
-I Install win32 service [win2k only]
-R Remove win32 service [win2k only]
-s use server named <server> in config file
-t read config from <file> (REQUIRED)
```

Changing the Router Desktop Install to Install as a Service

If for some reason you did not initially install the Event Router as a service, you can do so now by performing the following steps:

To install the service

1. Before you begin, log in as Administrator or as a user with Administrator privileges.
2. From the *Start* menu, select *Run>cmd* and click *OK*.
3. At the command-line, type *c:* and press *Return*. Type the letter of the drive on which you installed the Router. For purposes of this example, drive *C* is used.
4. Change to the directory where the Router is installed. For example,
`cd "\Program Files\KnowNow\Event_Router"`.
5. Type: `bin\nsd.exe -I-t conf\knrouter.conf`
6. After the service is installed, type `exit`.

To start the service

1. From the *Start* menu, select *Settings>Control Panel>Administrative Tools>Services>KnowNow Event Router-runtime*.
2. Click the *Properties* icon.
3. Select the *Recovery* tab.
4. In the *First failure* field, select *Restart the service* and click *OK*.
5. Click the *Start Service* icon. The word *Started* displays in the *Status* column.

To ensure the service is running

1. From the *Start* menu, select *Run>C:\Program Files\KnowNow\Event_Router\log\server.log*
2. Click *OK*.
3. Scroll to the bottom of the log and locate the line which contains the text: **listening on <IP address:port>**.
4. Note the IP address and port number, then close the window.
5. From the *Start* menu, select *Run* and type, **http://IP address:port**. Use the IP address and port number you noted in the previous step. For SSL, use **https:// IP address:port**).
6. Click *OK*.
7. The KnowNow Event Router startup page displays with instructions on how to run the tests.

To stop the service

1. From the *Start* menu, select *Settings>Control Panel>Administrative Tools>Services>KnowNow Event Router-runtime*.
2. Click the *Stop Service* icon.

To remove the service

1. From the *Start* menu, select *Run>cmd* and click *OK*.
2. At the command-line, type *C:* and press *Return*. Type the letter of the drive on which you installed the Router. For purposes of this example, drive *C* is used.
3. Change to the directory where the Router is installed, for example, `cd "\Program Files\KnowNow\Event_Router"`.
4. Type `bin\nsd.exe -R -t conf\knrouter.conf`.
5. After the service is removed, type `exit`.

Router Utilities

This chapter discusses the Router Utilities. The following sections are discussed:

- [Router Utilities](#)
- [Using the Utilities](#)
- [KN Config Utility](#)

Router Utilities

The following HTML-based utilities are installed with the KnowNow Event Router:

KnowNow Utility	Description
KN Test	Performs automated tests to ensure that the web browser, KnowNow Event Router, and the JavaScript Microserver are functioning properly.
KN Ping	Measures the event routing performance over a connection between the KnowNow Event Router and a web browser, returning data on round-trip latency and throughput.
KN Topic Modeler	Allows you to navigate through event topics and topic information. You can also manage topics, events, routes, access control, and filters on topics and routes. For more information, refer to the KnowNow Topic Modeler Guide.
KN Config	Allows a server developer/administrator to configure the Router.

Using the Utilities

When you start your KnowNow Event Router, the IP address and the port number it is using are displayed either directly to a console or to a log file (for example, 10.10.10.45 port 8000).

To run a utility:

1. Connect to the Router status page using a web browser, for example, **http://10.10.10.45:8000**.
2. Select the *Router Utilities* link; the page displayed provides links to each of the four utilities listed below.
3. Select a link; the utility displays in a new browser window.

To use KN Topic Modeler:	allows you to navigate through event topics and topic information. For more information, refer to the KnowNow Topic Modeler Guide.
To use KN Config	allows you to edit the Config file. For more information on this utility, refer to " KN Config Utility " on page 3-3.
To use KN Test	follow the instructions provided as part of the utility.
To use KN Ping:	enter a number in the first text field (or leave it at 100, the default) and click <i>Run Tests</i> .

For more information on utilities, and to download sample applications and components, visit the KnowNow developer site:

<http://developer.knownow.com>.

KN Config Utility

The KN Event Router Configuration Utility allows a server administrator/developer to configure the Router.

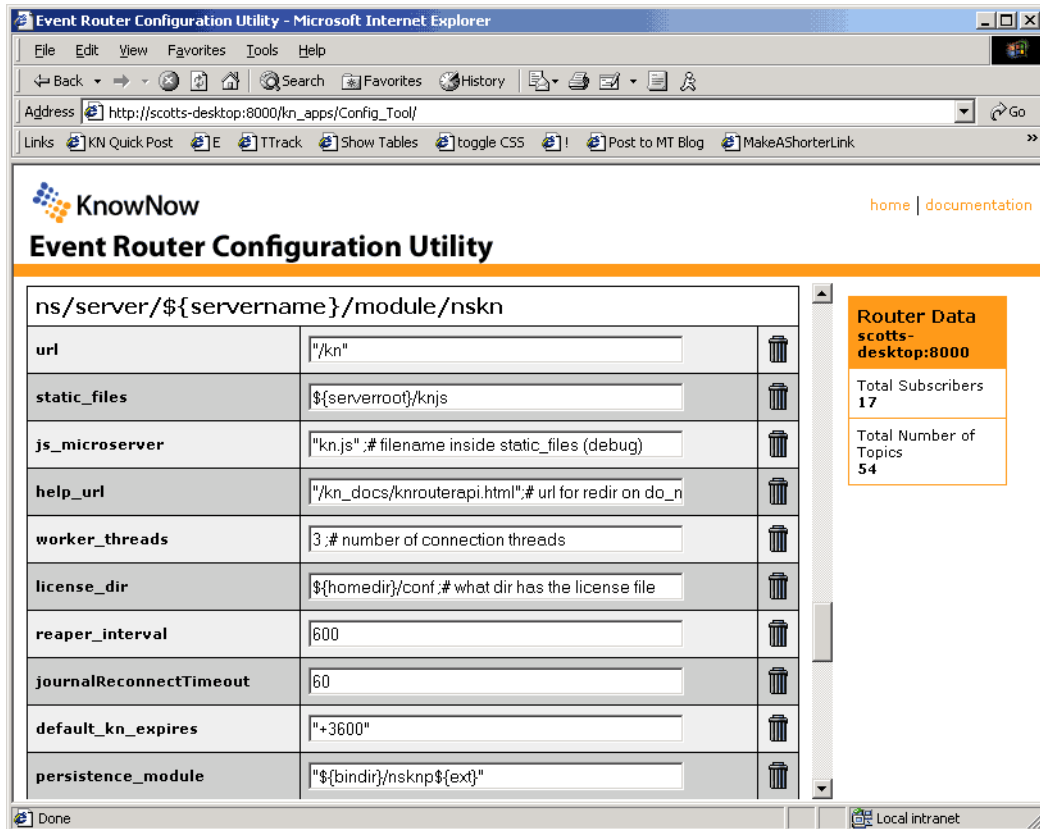
Starting the Config Tool

To start the Config Tool, use the following process:

1. Navigate to the URL where the tool is located The URL can be found on the Router Utilities startup page.
2. An **.htaccess** login prompt appears.
3. Login with root privileges.
4. If the login is successful, the main configuration screen appears. See Figure 3-1.

The Config Tool Screen

Figure 3-1. The Configuration Tool Screen



Note: On Windows platform, the Config Utility will re-start by itself only if the Router is running as a service.

The Config Tool screen is divided into the following three frames:

- Header: contains the KnowNow logo and links to the product documentation.
- Config Editor (left side): contains editable fields.
- Router Data (right side): provides information about the Router

Config Editor

The Config Editor screen display the following information in editable form:

- All constants declared in the config file
- all defined sections
- all parameters grouped by section

You can edit this information but the changes are not saved until you click the **Save Changes** button.

The Config Tool screen comprises of the following buttons:

Button	Description
Add a Constant	Allows you to add a constant.
Add a Parameter	Allows you to add a parameter to a section.
Add a Section	Allows you to add a new section.
Delete a Constant	Allows you to delete a constant.
Delete a Parameter	Allows you to delete a parameter.
Delete a section	Allows you to delete a section.
Save Changes	Prompts you with an alert that tells you that it will save the changes and restart the Router. You have the option to cancel the action.
Reset to Original Values	Returns all editable fields on the screen to their original values.

For more information on using these fields, refer to [“Using the Config Editor” on page 3-6.](#)

Router Data

The Router Data section of the Configuration screen provides the following information:

- total number of subscribers to the Router since the Router was first started.
- total number of topics created on the Router since the Router was first started.

The Router Data information comes out of `kn_statistics` and is updated to an interval specified in the `conf` file. The default interval is 10 seconds. Discarding the Router persistence files and restarting the Router resets these values to zero.

Using the Config Editor

The Config Editor can be used to edit, add, or delete a constant, to edit, add, or delete a parameter, and to add or delete a section.



Note: Config Utility cannot be used through a load balance interface in a cluster configuration. The recommended approach is that you get one version of the `.conf`, `perm`, `passwd`, and `group` files just the way you want it, copy these files to each of the cluster machines, and then restart the cluster machines.

Editing a constant

To edit a constant, do the following:

1. Click inside the text box that contains the constant value you want to edit.
2. Using your keyboard, change the value inside the text box.
3. Click **Save Changes** to make sure the changes take effect.

Adding a new constant

To add a new constant, do the following:

1. Click the **Add Constant** button at the bottom of the Constants section.
2. At the prompt, enter the name of the new constant.
3. Click **OK**.
4. At the next prompt, enter the value of the new constant. The default is an empty string.
5. Click **OK**.

Deleting a constant

To delete a constant, do the following:

1. Click the trashcan icon next to the constant you wish to delete.
2. To confirm deletion, click **OK** at the prompt. To cancel deletion, click **Cancel**.

Editing A parameter

To edit a parameter, do the following:

1. Click inside the text box that contains the parameter value you want to edit.
2. Using the keyboard, change the value inside the text box.

Adding a new parameter

To add a new parameter, do the following:

1. At the bottom of the appropriate section, click the **Add Parameter** button.
2. At the prompt, enter the name of the new parameter.
3. Click **OK**.
4. At the next prompt, enter the value of the new parameter. Click **OK**.

Deleting a parameter

To delete an existing parameter, do the following:

1. Click the trashcan icon next to the parameter you wish to delete.
2. To confirm deletion, click **OK**. To cancel deletion, click **Cancel**.

Adding a section

To add a new section to the Config Editor frame, do the following:

1. Scroll to the bottom of the Config Editor frame and click the **Add Section** button.
2. At the prompt, enter the name of the new section.
3. Click **OK**.
4. The new section appears at the bottom of the Config Editor frame.
5. To add parameters to the new section, follow the steps for adding parameters above.

Deleting a section

To delete an existing section from the Config Editor frame, do the following:

1. At the bottom of the section that you want to delete, locate and click the **Delete Section** button.

2. Deleting a section also deletes all parameters associated with it. To confirm deletion, click **OK** at the prompt. To cancel, click **Cancel**.

Saving the Edited Config Data

When you finish editing the config data, you must save your changes and send them back to the Router. The Router will attempt to restart with the new settings.

1. Scroll to the bottom of the Config Editor frame and click the **Save Changes** button.
2. The changed config data is sent to the Router. The Router then attempts to restart with the new settings.
3. If the attempt is successful, you get a notice saying that the Router has restarted. Click **Edit Configuration** if you need to edit the config data again.
4. If the restart fails, you get an error message. Contact the system administrator for help.

Restoring the Config Data

You can restore configuration changes only if the **Save Change** button has not been clicked. To restore the config data to its last saved state:

1. Scroll to the bottom of the Config Editor frame, and click **Revert to Saved** button.
2. The Config Editor reloads the last saved configuration data from the Router. All unsaved changes you made to the config data are lost.

Exiting the Config Tool

There is no logout process to exit the tool. Simply navigate to another URL.

Managing the KnowNow Event Router

This chapter discusses management of the KnowNow Event Router. The following sections are discussed:

- [Access Control in the KnowNow Event Router](#)
- [Clustering](#)
- [Persistence](#)
- [SSL Configuration](#)

Access Control in the KnowNow Event Router

The KnowNow Event Router allows administrators to control:

- which users can publish events to a particular topic
- which users can subscribe to a particular topic
- which users can create subtopics for a given topic

Configuration of the access control mechanisms takes place in three files:

- `passwd` - defines valid users and their passwords
- `group` - defines groups of users
- `perms` - defines the permissions given to specified users and groups

These three files can be found in the **conf** subdirectory of the KnowNow Event Router's install directory. See "[Authentication/Authorization modules](#)" on page 4-5" for parameters that must be edited to ensure authentication and authorization work properly.

The three access control files are discussed in more detail below.

passwd

Entries in the **passwd** file are similar to UNIX/etc/passwd entries. The first field contains a username and the second field contains the `crypt()` encrypted password for that user. The fields are separated by colons. Note that five (5) colons can optionally follow the encrypted password for compatibility with UNIX password editing tools. For example:

```
steve:ubphBI51DzSYc:::::
```

You need to use the `passgen` utility to create encrypted passwords. The `passgen.exe` is located in the `/bin` subdirectory. You give it a username and password on the command line and it spits out the encrypted entry for the `passwd` file. For example,

```
$ passgen steve foobar
```

will give

```
steve:aa9TsAg8bf3eo
```

which can be copied into the `passwd` file by the administrator.

group

Entries in the `group` file are similar to UNIX/`etc/group` entries. The first field specifies a group name and the second field lists the users that are part of the group. The list of users in the second field must all be defined in the `passwd` file. Three colons separate the group name from the list of users in that group to maintain compatibility with UNIX group editing tools; comma delimiters are used to separate users. See the example below:

```
eng:::nsadmin,nobody
```

perms

This file defines the permissions given to users and groups on the KnowNow Event Router. There are four commands valid in this file:

allowuser	defines permissions associated with a single user
allowgroup	defines permissions to a group of users
denyuser	denies access to a specific user
denygroup	denies access to a specific group of users

The parameters of the **perms** file have the following format:

- ACTION—allowuser, allowgroup, denyuser, denygroup
- INHERITANCE—inherit or noinherit, default is inherit
- METHOD—GET, POST, topic, notify, route
- URL—the path on the KnowNow Event Router
- ENTITY—either the name of a user or group, as specified in the **passwd** and **group** files

For Example:

To allow only user “jenkins” to GET or POST to the Router:

```
allowuser inherit GET /kn jenkins
allowuser inherit POST /kn jenkins
```

To deny only user “steve” to GET or POST to the Router:

```
denyuser inherit GET /kn steve
denyuser inherit POST /kn steve
```

To restrict the topic **/foobar** such that only one user can publish to the topic, modify the code as follows. In this example, *sophie* is the only user who will be able to publish to the specified topic:

```
allowuser noinherit notify /kn/foobar sophie
```

To restrict the topic **/foobar** such that only one user can delete or modify the properties of the topic, modify your code as follows. In this example, *steve* is the only user who will be able to delete or modify the specified topic.

```
allowuser noinherit topic /kn/foobar/kn_routes steve
```

To restrict the topic **/foobar** such that only one user can add or edit routes out of a specific topic, modify your code as follows. In this example, *olivia* is the only user who will be able to add/edit the routes out of a specified topic.

```
allowuser noinherit route /kn/foobar/ olivia
```

The URLs in the `allowuser` lines are the topic names prepended with the path on the server the Router is connected to. So in the default case, where **nomad** is listening on **/kn**, you prepend that to the topic names as above.

In addition to the **allowuser** parameter, you can also use **allowgroup** to restrict access to only specified groups. The **denyuser** and **denygroup** parameters can also be used to subtract permissions that have been allowed by the **allowgroup** parameter setting.

passwd file entries are **htaccess-style** passwds; only the first two fields are used by the server. You can specify that a user not have a password by leaving the second field blank. In this case, the user still has to give a username to access the topic, but can leave the password field of the http challenge blank.

Authentication/Authorization modules

Before you can use the authentication and authorization modules, make sure you use a text editor to uncomment the following lines in `knrouter.conf` and verify the setup of the authentication and authorization files:

```
#
# Load Authentication module?
# ns_param authentication_module "${bindir}/knauthenticate${ext}"

#
# Load Authorization module?
#ns_param authorization_module "${bindir}/knauthorize${ext}"
```

After you have uncommented the above lines, ensure that the following settings are uncommented and point to the correct files:

```
# Authentication module settings
ns_section "ns/server/${servername}/module/knauthenticate"
ns_param passwordFilePath "${homedir}/conf/passwd"

# Authorization module settings
ns_section "ns/server/${servername}/module/knauthorize"
ns_param permsFilePath "${homedir}/conf/perms"
```

Using Authentication and Authorization to Restrict Access

KnowNow's Event Router ships with default Authentication and Authorization modules. The Authentication module uses Unix-like password file to store a list of users and their associated passwords. For more information on passwords, refer to ["passwd" on page 4-2](#).

The Authentication module is invoked to validate a user's credentials. The User ID and password are stored in the `"passwd"` file in the `"conf"` directory of the Event Router installation. An example of the `"passwd"` file is shown below:

```
# passwd file for KnowNow authentication module
#
# format is
#      username:passwd
# password should be encrypted using standard unix crypt() style, and
# may be generated with 'passgen' utility.
jenkins:aamt6nQwlvkgE
"":
guest:aaLmYvG0jXYCk
knadmin:aaaJBvWPq15xU
```

In the example above, four users have been identified. The users `"jenkins"`, `"knadmin"`, and `"guest"` have passwords associated with them; however, the user `""` refers to any unauthenticated user.

In addition to the `"passwd"` file, another file is used to store a list of groups and their associated users. The `"group"` file is also located in the `"conf"` directory. For more information on the group file, refer to ["group" on page 4-3](#). The `"group"` file may look like the example below:

```
#
# This is a sample group entry. The format is
# <group name>:<userid>,<userid>, ...,<userid>
#
#users:steve,jenkins,rga,superman
all:"",jenkins,guest,knadmin
admins:jenkins,knadmin
```

In the example above, two groups are defined, where `"all"` has all four users, but the `"admins"` group only has `"jenkins"` and `"knadmin"`.

The Authorization module uses the `"perms"` file to read user privileges. For more information on the `"perms"` file, refer to ["perms" on page 4-3](#).

The KnowNow Event Router requires all users to provide valid credentials, which can be implemented using the following lines in the `"perms"` file:

```
#
# Add this entry to lock down the router and require all users to
login.
#
denyuserinheritGET      / ""
denyuserinheritPOST    / ""
```

If a browser is used to access the Event Router, then the browser will pop-up its login window for user ID and a password.

Restricting Topics to Authenticated Users only

The following entry in the “perms” file restricts access to a specific topic, such that only authenticated users can access the topic:

```
#
# These entries lock down a topic in the router so that only
authenticated users may access it.
#
denyuser inherit route /kn/what/test ""
denyuser inherit topic /kn/what/test ""
denyuser inherit notify /kn/what/test ""
```

The following entry restricts access to a specific topic, such that only the user “jenkins” can access it:

```
#
# These entries lock down a topic so that only the user “jenkins” can
use it.
#
allowuser inherit route /kn/onlyone jenkins
allowuser inherit topic /kn/onlyone jenkins
allowuser inherit notify /kn/onlyone jenkins
```

Restricting the `kn_config` topic

The `/kn_config` topic holds an event that reflects the contents of the “`knrouter.conf`” file. New events posted to this topic will cause the contents to be replaced by the data in the new event’s “`kn_payload`” header.



Caution: System administrators will want to lock down this resource, since new events posted to this topic, `/kn_config`, could restart the Router.

The following entries can be used to restrict access to `/kn_config` to “admins” only:

```
#
# Protect the /kn/kn_config topic so that only the users in the admin
group can use it.
#
allowgroup inherit route /kn/kn_config admins
allowgroup inherit topic /kn/kn_config admins
allowgroup inherit notify /kn/kn_config admins
```

Restricting Access to Various Event Router Operations

This section describes how you restrict access to various operations in the Router: deleting a topic, attaching filters to a topic, attaching filters to a Router, fetching and uploading a config file, setting default event expiration times, and setting max queue lengths.

Deleting a topic

Any user that has permission to add or edit a topic’s properties also has permission to delete the topic. You can restrict deletion of a topic by using the permissions file settings. For more information on the permissions file, refer to [“perms” on page 4-3](#). Some topics may not be deleted. The Event Router specifically protects against anyone deleting:

- `/`
- `/kn_config`
- `/kn_statistics`
- `/kn_filters`

Attaching filters

Filters can be added to any topics, except the subtopic topics. The following apply:

- A filter cannot be added to any topic containing `/kn_subtopics`
- Filters can be attached to special topics, such as `/kn_config` and `/kn_statistics` during the runtime of a Router; however, the filters will not be persisted.

Fetching and replacing the contents of the config file

- You can fetch a copy of the config file by retrieving the `config_file` event in the `/kn_config` topic. The payload of the `config_file` event contains the contents of the Router configuration file.
- You can replace the contents of an existing config file by posting a new `config_file` event to the `/kn_config` topic. The event must contain the new configuration file within the payload of the event. When the Router receives the event, it will:
 - make a copy of the existing config file and write it to disk
 - write the new config file (from the payload of the new `config_file` event to disk
 - restart itself with the new configuration file.



Note: The Router must be running without the `-f` option (on Solaris/Linux), or as a service on Windows to restart itself automatically. For more information on Router installation, refer to [“Installing the KnowNow Event Router” on page 2-2](#).

Setting default expiration time

You can use the `kn_default_kn_expires` header to set a default expiration time for events within a topic. Events expire based on the following rules:

- `/kn_subtopics` topics always have an event expiration time set to infinity
- `/kn_routes` topics default to having an event expiration time of infinity
- `/kn_journals` topics default to having an event expiration time of `JournalReconnectTimeout` (which defaults to 60 seconds). The `JournalReconnectTimeout` parameter can be specified in the configuration file.

To define a maximum amount of time a journal can be set to maintain events, use the `maxJournalReconnectTimeout` parameter in the config file. You may change the amount of time that a journal stores its events by setting the `kn_default_kn_expires` property on a journal topic. The default value of `maxJournalReconnectTimeout` is set to 3600 seconds (1 hour).

- All other topics default to having an expiration time of `default_kn_expires` (which defaults to 3600 seconds). The `default_kn_expires` parameter can be specified in the configuration file.
- The event expiration time of any topic (except `/kn_subtopics` topics) can be changed by using the `kn_default_kn_expires` header. You can set this parameter using the `set_topic_property` Router command.

The following example will set the topic called “*mytopic*” to have a default event expiration time of 600 seconds:

```
do_method=set_topic_property;kn_to=/mytopic;kn_default_kn_expires=600
```

Setting maximum queue length

You can limit the maximum number of events that are contained within a topic at any given time by setting the `kn_max_queue_size` property on the topic. By default, topics can have an unlimited number of events. If the `kn_max_queue_size` parameter is used, the topic acts as a FIFO (first in, first out) queue where only the last *n* events will be stored. For instance, the following example will set a max event queue length of 50 in the topic “*mytopic*.” If the topic has 50 unexpired events, and a new event arrives, the oldest event in the queue will immediately expire in order to make room for the new event:

```
do_method=set_topic_property;kn_to=/mytopic;kn_max_queue_lenght=50
```

Clustering

What is Clustering?

A cluster is when two or more Routers are configured into a cluster to provide:

- machine fail-over mechanism
- Scalability of connections

All Routers in a cluster mirror each other to look and act like one Router.



Note: Clustering Routers does not help improve performance or throughput. In fact, there is a slight amount of overhead associated with clustering which can cause a slight decrease in performance.

Failover Mechanism

The Failover mechanism allows for uninterrupted Router requests to happen even if one of the Routers in the cluster goes down. For example, if a cluster contains Router A, Router B, and Router C, and Router A in the cluster goes down, the other Router(s) continues to serve Router requests without interrupting service. However, you need to be careful when clustering machines to ensure data integrity.

Since there is no guaranteed ordering within an asynchronous distributed system such as a Router, it is important to understand how the cluster works to ensure that data remains in sync across all nodes in a cluster.

If you have multiple clients or an asynchronous client publishing update events to topics, you should feed all requests for those clients to a single node in the cluster. Otherwise, events crossing paths between cluster nodes will lead to inconsistency across the cluster.

For instance, if publishers P1 and P2 are both sending updates to a cluster with stock ticker and price information on IBM, the following situation could occur. Assume that P1 sends an event to node N1 with `kn_id="IBM"` and `kn_payload="100"`. At the same time, P2 sends an event to node N2 with `kn_id="IBM"` and `kn_payload="102"`. Each node will accept the request and pass it along to its peers. But due to latency issues, and the fact that the events are being duplicate-squashed and updated, it is possible

that node N1 could end up with the opposite event as node N2. The current solution for this problem is to feed all incoming cluster requests to the same node. This can be accomplished using a load balancer. If one node goes down, the load balancer can switch to using the other node, and thus prevent the loss of routing service.



Note: If a machine in the cluster goes down, and you restart the machine, it does not make any attempt to re-sync with the other machines. There is no transaction mechanism built into clustering; therefore, events (at any given time) may appear to be in a slightly different order. Ordering of events is not preserved across the nodes in a cluster. Therefore, topics may contain the same events, but they may be stored in a different order on each clustered Router.

Scalability of Connections

Clustering can also be used to increase the total number of connections to the Router. The number of connections on a Router is limited to the operating system's limits for open file descriptors. Solaris and Linux allow 64K descriptors; whereas, Microsoft Windows allows up to 2K file descriptors. You can exceed the operating system's limits by running several Routers in a cluster and distributing the connection load over multiple machines.

Router requests can be directed to any node in the cluster. The cluster module automatically takes each request and mirrors it across all of the other machines in the cluster. Therefore, every node in a cluster has the same topic hierarchy and receives all of the same events. However, the nodes in the cluster are not guaranteed to be exactly alike. The cluster configuration and route topology must be carefully planned to ensure that cluster nodes have the required level of similarity. For more information, refer to ["How to Cluster?" on page 4-13](#).

How to Cluster?

To run a cluster, you must first configure the **cluster** parameter, which is located in the **nsknc** section of the **KNRouter** configuration file. Before you can configure the cluster, you may need to uncomment the **nsknc** parameter in the modules section of the `knrouter.conf` file. The cluster parameter takes a list of <host>:<port> combinations, separated by white space. There is no limit to the number of peers in the cluster, although practically 64 should be the limit. The default value is a dummy value that will not create any clusters.

The following example presents a scenario for setting up a cluster of two machines. See figure below:

1. Set the `hostname` to be the URL of your local site. This section will be the same in the config files for both the machines in the cluster. Do NOT include the `hostname` in the cluster parameter of the config file.

```
set hostname http://www.mysite.com
```

2. Set the `local_host` in the cluster parameter of the config file.

For machine 1:

```
ns_section "ns/server/${servername}/module/nskn"
local_host foobar.com:8888:443
```

For machine 2:

```
ns_section "ns/server/${servername}/module/nskn"
local_host 1.2.3.4:8080:443
```

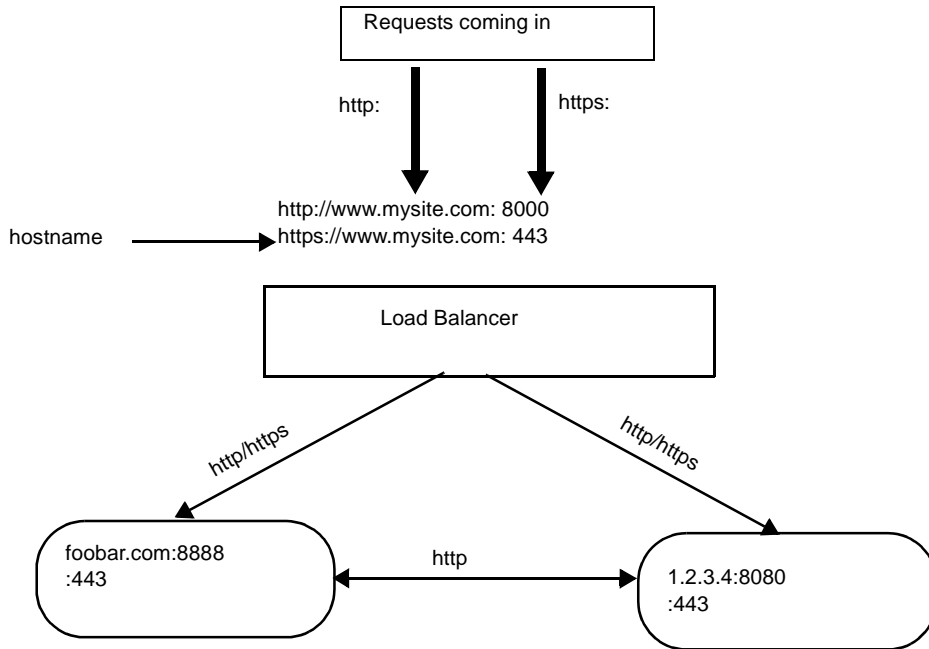
If you copy the config file from one machine in the cluster to another, make sure that the `local_host` statement is set up for that specific machine and not copied over for a different machine.

3. Set the cluster statement as follows:

```
ns_section "ns/server/${servername}/module/nsknc"
ns_param cluster "\
1.2.3.4:8080 \
foo.bar.com:8888 \
"
```

Notice that the example defines a cluster made of two peers: one listens on IP address 1.2.3.4 at port 8080; the other, which resolves to the IP address the DNS

returns for **foo.bar.com**, listens on port 8888. Notice also that quotes are necessary; however back slashes (\) are used to break lines for a cleaner code.



Note: The config file for each machine in the cluster should be exactly the same except for the local host statement.

Important Clustering Considerations:

- On each client cluster box, remember to set the hostname to the externally visible DNS name of the cluster.
- You must keep the <host>:<port> combinations in sync with the top-level “**httpport**” and “**address**” variables in the configuration file. Each peer must be able to recognize itself as one of the values in the cluster list. Also, the client must ensure that the **nsknc** module is loaded at **KNRouter** run-time. The existing entry for **nsknc** in the “**modules**” section is commented out by default.
- Each value in the cluster list must contain the exact same cluster value in their local copy of the configuration file.
- When clustered, the cluster can receive SSL requests but the intercommunication between clustered machines is only http. To ensure security, make sure that all machines within the cluster are behind the same firewall. Clustering requires HTTP communications to be enabled so that cluster members can communicate. If you need HTTPS-only Routers to work in a cluster, use a firewall or load balancer to block connections to port 80 on the cluster boxes (and only allow HTTPS connections on port 443).
- The KnowNow Config Utility cannot be used through a load balance interface in a cluster configuration. The recommended approach is that you get one version of the `.conf`, `perm`, `passwd`, and `group` files just the way you want it, copy these files to each of the cluster machines, and then restart the cluster machines.

Persistence

The KnowNow Event Router uses Berkeley's SleepyCat database (see <http://www.sleepycat.com>) to maintain a persistent state on disk. This allows the Router to be stopped and then restarted in the same state (that is, all of the topics and events that exist when a Router exits are available when the new Router starts).



Note: Since there is no guaranteed ordering within an asynchronous distributed system such as a Router, events (at any given time) may appear to be in a slightly different order.

Previous versions of the KnowNow Event Router (v1.2 and earlier) used a custom implementation to achieve this functionality. The v1.3 Router had an embedded implementation of the SleepyCat database to maintain persistent data. The v1.5 Router has a new dynamic SleepyCat module to maintain persistent information. The v1.5 Router is almost identical to the v1.3 module except for a few minor differences to the configuration file.

Important:

- Version 1.7 persistent files are different than the earlier versions. The Router will automatically upgrade to v1.7 when it's run. You cannot use the 1.7 database on a 1.6 Router after it has been upgraded.
- Old persistent files (v1.2 and earlier) are not backwards compatible with the v1.3 or v1.5 Router formats. Starting a v1.3 or v1.5 Router will automatically delete old persistence files (v1.2 or earlier).
- v1.3 versions of the persistence files will automatically be upgraded to use the v1.5 Router format the first time you use the v1.5 Router. However, you will NOT be able to roll back to the v1.3 format once the Router has upgraded the database.

The v1.5 Router supports the following configuration parameters for persistence. In the configuration file `conf/knrouter.conf`:

Under the KnowNow Event Router Persistence -- nsknp:

```
ns_section "ns/server/${servername}/module/nsknp";
ns_param persistence_file ${serverroot}/knrouter.prs;
ns_param persistence_interval 600;
```

Under the KnowNow section (nskn):

```
ns_section "ns/server/${servername}/module/nskn";
ns_param persistence_module "${bindir}/nsknp${ext}"
```

persistence_file

The `persistence_file` parameter specifies the location of the database file. If the file does not exist when the Router starts, it is created automatically. A temporary database directory is also created automatically, if it does not exist. The database directory is only needed while the Router is running and may be deleted by the Router or manually after the Router is shut down. The temporary database directory has the same name as the persistence file with the string `_db` appended. For example, when the Router is running you may see something like this:

```
/usr/local/knrouter/runtime/knrouter.prs//Database file  
/usr/local/knrouter/runtime/knrouter.prs_db//Database directory
```

persistence_interval

The `persistence_interval` parameter specifies how often non-atomic persistence information will be flushed out to disk. The interval is specified in seconds. In the example above, the Router flushes a snapshot of its state to disk every 600 seconds. This means that if the Router is abnormally terminated or crashes, it will only lose 600 seconds (at most) of information that cannot be recovered. The Router will always try to persist its state when it exits, so it is possible that even if the Router crashes, 100% may be recoverable.

persistence_module

The `persistence_module` parameter specifies the name of the persistence module. The persistence functionality of the v1.5 Router has been moved out of the Router core and into a module. This allows you to develop your own modules for handling persistence using the Router's persistence API functions. The default configuration includes a module for using the SleepyCat database for persistence. You may configure the Router to not persist any information about the Router by removing this statement from the configuration file.

atomic_router

The `atomic_router` parameter instructs the Router to flush all topic and event information to disk continuously (as events occur). This allows the Router to have the best chance to recover 100% of its state following a shut down or crash, but may have a slight performance penalty.

atomic_topic

The `atomic_topic` parameter instructs the Router to continuously save a single topic's events to disk as they occur. This works in the same way as the `atomic_router` parameter, but only for one topic (per `atomic_topic` statement). Multiple topics can be run in atomic mode (without the whole Router being atomic) by specifying each atomic topic name with an `atomic_topic` statement in the configuration file.

Moving Persistence Information

The new persistence information **should not be moved using regular file move/copy/rename commands**. Using these commands could result in two potential problems:

- if the Router is running, it is likely that the copy of the database will be inconsistent from the original or could even be corrupted (because the Router could be in the middle of writing while you are copying).
- the database actually stores file-specific information within the database that will become inconsistent if you copy, move, or rename the database using standard file system commands, even if the Router is not running.

To get around this problem, use the `db_kncopy` utility to copy Router persistence information. This utility must be used every time you copy, move, or rename the persistence information. In addition, it can be used while the Router is actually running. The utility actually connects to the database, sets the necessary locks, and allows you to backup your persistence database without stopping the Router. The syntax of the command is:

```
db_kncopy [-V] source_db destination_db
```

`-v` prints version information

`source_db` is the source database persistence file

`destination_db` is the destination database persistence file

The following example copies the `knrouter.prs` file to the `backup.prs` file .

```
db_kncopy knrouter.prs backup.prs
```

SSL Configuration

You can turn on SSL support for the KnowNow Event Router by setting options in the config file. To turn on SSL support:

1. In the config file, uncomment the following line by removing the # sign:

```
# ns_param nsopenssl ${bindir}/nsopenssl${ext}
```

Note that this line loads the **nsopenssl** module; it is commented out by default when you install the KnowNow Event Router.

There is a new section in the config file, **nsopenssl**, which holds the configuration directives:

```
ns_param ServerPort           $httpsport
ns_param ServerHostname      $hostname
ns_param ServerCertFile      ${homedir}/conf/cert.pem
ns_param ServerKeyFile       ${homedir}/conf/rsa.pem
```

Parameter	Definition
port	configured from the httpsport setting, common to the entire KnowNow Event Router. The httpsport setting is located near the top of the configuration file; it sets the port on which to listen for SSL-encrypted connections. The default is 8443.
hostname	configured from the hostname setting, common to the entire KnowNow Event Router.
CertFile	should point at the signed certificate file from the Certification Authority (CA). The KnowNow Event Router certificates need to be in PEM format, which can be generated by several public CAs, as well as by all the major CA infrastructure products. This file resides in the conf/ subdirectory, along with the configuration file.
KeyFile	should point at the RSA private key for the Router. This file resides in the conf/ subdirectory, along with the configuration file.

Generating and Installing Certificates

To enable SSL on a Router, you must generate and install a certificate. To do this:

1. Generate the RSA key for the Router by running the **genrsa** utility, located in the **bin/** directory. Run the command as follows:

```
$ /usr/local/kn/bin/genrsa -out /usr/local/kn/conf/rsa.pem
```

After running the command, an RSA key in PEM format is placed in your **/usr/local/kn/conf** directory.

2. Generate a Certificate Signing Request (CSR) by running the **req** utility, located in the **bin/** directory.

```
$ /usr/local/kn/bin/req -new -key /usr/local/kn/conf/rsa.pem
```

Type the appropriate information into the data fields:

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgets Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:
```

Note: In the Common Name field, type the full **hostname** of the machine on which the Router is running. For example, **www.mycompany.com**.

3. When finished, the **req** utility prints a certificate request. Send this request to the appropriate CA and complete their approval process to receive a signed, valid certificate.
4. Place the valid certificate in the **usr/local/kn/conf/cert.pem** file and start up the KnowNow Event Router.

Tuning Parameters

This chapter discusses tuning parameters for the KnowNow Event Router. The Event Router parameters and their descriptions are provided in [Table 5-1](#). Where applicable, the default values are also given.

KnowNow Event Router Tuning Parameters

The following table provides a list of KnowNow Event Router parameters that you can set:

Table 5-1. KnowNow Event Router Tuning Parameters

Event Router Parameter	Description
address	IP address the Router should bind to. This parameter is useful if a machine has more than one network connection. Default value: [ns_info address], which tells the Router to figure out the address by itself.
atomic_router	If True, every event transmitted to the Router will force the Router to persist state to disk. This incurs a significant performance penalty. Default value: False
atomic_topic	Allows a single topic to be configured as atomic, in which case event transmissions to that topic force the Router to persist state to disk immediately. Can be used multiple times to make topics atomic.
authorization_module	Authorization modules are called to verify access rights, to see if a particular set of credentials can access a particular URL on the Router.
authentication_module	Authentication modules are called to verify a user, usually based on <code>userid</code> and <code>passwd</code> . The Router never calls the Authentication module directly, it is up to the authorization layer to call it using <code>knAuthenticateUser</code> .
cluster	Enables a set of machines to participate in a cluster. Default value: dummy value, no clusters are created.
kn_default_kn_expires	The length of time an event lives if an expiration time is not set. This value can be set to "infinity", which means the event will live forever. Default value: +3600 seconds (1 hour)

Table 5-1. KnowNow Event Router Tuning Parameters

Event Router Parameter	Description
filter_module	Multiple filter module statements are allowed. A module can be attached to a topic or a route and filter events based on their <code>kn_payload</code> content with a regular expression that has been assigned to the filter. A filter module can handle the following four filter functions for events that match the regular expression (deny, allow, reroute, and transform). Filter module for the KnowNow Event Router.
help_url	URL to redirect users to when they perform a do_method=help on the Router.
hostname	The name of the host that the Router is running on. Default value: <code>[ns_info hostname]</code> , which tells the Router to figure out the hostname.
httpport	Specifies the port that the Router listens on for requests. Default value: 8000
javascript_domain	Domain to set on all returned javascript. Used in cross-domain Router setups. Refer to the <i>Writing Cross-Domain KnowNow Web Applications</i> document for more detail.
javascript_kn_server	kn_server setting for use in cross-domain Router setups. If cross-domain scripting is turned ON, the URL specified for this parameter MUST be the one used by the browser to connect to the Router. Refer to the <i>Writing Cross-Domain KnowNow Web Applications</i> document for more detail.
journalReconnectTimeout	The length of time that journals live after connections close. If clients reconnect in the time period specified in this parameter, they can recover connections atomically. Default value: 60 seconds
js_microserver	Filename inside static_files directory of the Javascript Microserver.
license_dir	The directory that contains the license file for the Router. Default value: <code>conf</code> subdir of Router.
local_host	Aliases for the machine that the Router is running on. This can include various combinations (machine name: port number; IP address:port number) that identify the machine as the host machine on which the Router is installed.

Table 5-1. KnowNow Event Router Tuning Parameters

Event Router Parameter	Description
maxconnections	Controls the maximum number of simultaneous connections the Router can handle. Default value: 100
maxkeepalive	Controls the maximum number of requests that can use the HTTP Keep Alive . If a value is not set, this parameter uses the values specified in maxconnection .
maxJournalReconnectTimeout	Limits the maximum amount of time that a journal can be set to maintain events. You may change the amount of time that a journal stores its events by setting the <code>kn_default_kn_expires</code> property on a journal topic. Default value: 3600 seconds (1 hour)
maxline	Maximum HTTP GET request size in bytes. Default value: 8K
maxpipethreads	Controls the number of threads which are responsible for reading in HTTP and HTTPS requests. Once a pipethread parses a complete HTTP/HTTPS request, it is handed off to a worker thread (controlled by the <code>maxthreads</code> parameter). Default value: 3
maxport	Maximum size of event to accept in bytes. Default value: 2MB
maxthreads	Controls the number of inbound simultaneous requests the Router can process. Note that this does not affect persistent connections; it affects the pool of threads used only to process incoming HTTP GET and POST events. Default value: 50
persistence_file	The filename to persist the Router state to. Default value: knrouter.prs
persistence_interval	How often, in seconds, does the Router persist its state. Default value: 600 seconds
persistence_module	Specifies the name of the persistence module. You can configure the Router to not persist any information about the Router by removing this statement from the configuration file. Default: Default configuration includes a module for using the SleepyCat database for persistence.

Table 5-1. KnowNow Event Router Tuning Parameters

Event Router Parameter	Description
reaper_interval	<p>How often, in seconds, does the Router look for and delete expired events. Set this value to be lower than the default value for large number of expiring events.</p> <p>Default value: 600 seconds</p>
static_files	<p>Location on disk that contains the Javascript Microserver and supporting files.</p>
url	<p>URL prefix to which the Router responds to requests.</p> <p>Default value: "/kn"</p>
tunnel_header	<p>The <code>tunnel_header</code> option allows you to send additional http headers down a tunnel connection when it is opened. You can set as many <code>tunnel_header</code> statements as you like. The <code>tunnel_header</code> parameter(s) must be set in the <code>nskn</code> section of the config file. For example:</p> <p>"proxy-hint: unbuffered"</p>
tunnel_threads	<p>The number of threads servicing connected clients. Set this number higher than the defaults for larger numbers of connected clients.</p> <p>Default value: 3</p>

Frequently Asked Questions

This section discusses the most frequently asked questions regarding the KnowNow Event Router, its configuration, installation, and other usage scenarios. The frequently asked questions are also listed on the developer.knownow.com.

Frequently Asked Questions

- Q. How can I make an SSL-accelerated KnowNow Router work?
- A. The KnowNow Config Utility may not work in a cluster, so you may have to modify the `knrouter.conf` file directly using a text editor.

In the `knrouter.conf` file, for the given hostname, add two `ns_param local_host` lines. For example, if the Router is at `myrouter.knownow.com`, add:

```
ns_param hostname      "myrouter.knownow.com"
```

and

```
# optimize router by specifying known local hosts
ns_param local_host    "https://myrouter.knownow.com:443"
ns_param local_host    "https://myrouter.knownow.com"
```

Note that the static content redirects for directories missing their trailing slashes may redirect to the wrong protocol. This is a known issue and fairly difficult to fix.

- Q. How do I run my Router on port 80?
- A. Disable any other web server (such as IIS or Apache) using port 80 or configure it to use a port other than 80. Then in your `knrouter.conf` file, set `httpport` to 80.

Restart the Router. If you are on Solaris or Linux, you will need to start the Router as a "root" user and use the "-u" command line option to choose a user other than "root" to run the Router.

- Q. How do I configure a cross-domain Router for ports other than 80?
- A. You can have the application server and the Router on the same IP address, for example, with the application server using port 80, and the Router using port 8000.

The `src` attribute of the `<script>` tag should contain the URL of the Router including fully-qualified *domain name* and *portnumber*. If you are not part of a domain, use the fully-qualified *machine name* instead.

```
<html>
<head>
<title>My KnowNow Application</title>
<script type="text/javascript"
src="http://kn.biz.com:8000/kn/?do_method=lib"><script>
</head>
```

To enable cross-domain communication between your application and the Router, you must modify the `javascript_domain` and `javascript_kn_server` parameters

in the Router's `knrouter.conf` file. However, only the `javascript_kn_server` needs to have the port number included with it.

For example, if the Router is running on port 8000,

```
ns_param javascript_domain "biz.com"  
ns_param javascript_kn_server "http://kn.biz.com:8000/kn"
```

Note that the `javascript_domain` does not include a port. Also note that this configuration will not work with Netscape 4.x browsers, because Netscape 4.x requires both the application server and the Router to use HTTP on port 80 or HTTPS on port 443.

Q. How do I restrict requests to the Router by IP address?

A. In the `KnowNow/Event_Router/conf` directory, edit the `passwd` and `perms` files. There's documentation inside these files that describes how you can use the files for denying or allowing access. See also, "[perms](#)" on page 4-3.

Q. How does the Router let you fetch and replace a config file?

A. There is a topic `/kn_config` that contains one event with `event_id="config file"` and in the `kn_payload` of that event is the config file. When you republish that event, it restarts the Router. (This also happens if you use the KnowNow Configuration Utility tool.)

Q. What is the maximum message size that the Router can handle?

A. This is controlled by the following line in the `knrouter.conf` file:

```
ns_param maxpost2097152 ;#max post size of 2mb (sed)
```

This line sets the maximum HTTP POST size, and therefore does not exactly correlate with payload size of a single event. For example, headers take up some bytes, and if you use the batching API, several events get stuffed into a single POST, etc.

Q. How can I use my browser to send commands to the Router?

A. It is sometimes useful to use your browser to send commands to the Router; for example, when debugging. To do this, all you have to do is browse to a URL that contains a command. Here are some examples:



Note: In the examples below, replace "router" with the hostname where your Router is hosted. Example, "host1.mycompany.com"

- Publish an event containing the payload "100" to the topic "/what/prices/ibm"

`http://router/kn/what/prices/ibm?do_method=notify;kn_payload=100`

or

`http://router/kn?do_method=notify;kn_to=/what/prices/ibm;kn_payload=100`

- Create a route between topicA and topicB

`http://router/kn/topicA?do_method=route;kn_to=/topicB`

or

`http://router/kn?do_method=router;kn_from=/topicA;kn_to=/topicB`

- A journal is a topic that ends in `kn_journal`. It represents a particular user's long-lived connection to the Router. A subscription is just a route from the subscribed-to topic, into the journal. So, to subscribe the user at `/who/anonymous/s/10023/kn_journal` to the topic `/what/prices/ibm`:

`http://router/kn/what/prices/ibm?do_method=route;kn_to=/who/anonymous/s/10023/kn_journal`

or

`http://router/kn?do_method=route;kn_from=/what/prices/ibm;kn_to=/who/anonymous/s/10023/kn_journal`

- Same as above, but with `do_max_n` of 10

`http://router/kn/what/prices/ibm?do_method=route;kn_to=/who/anonymous/s/10023/kn_journal;do_max_n=10`

or

`http://router/kn?do_method=route;kn_from=/what/prices/ibm;kn_to=/who/anonymous/s/10023/kn_journal;do_max_n=10`

Q. I am using a JavaScript Microserver to develop a chat application. How can I know from JavaScript when the KnowNow server is down? Is there a callback function or event to tell the application that the KnowNow server is down?

A. If the Event Router is down when you first attempt to fetch the application, then the Microserver library will be missing. A test in your application can reveal this:

```
if (! self.kn) alert("We are experiencing technical difficulties. Please wait a bit and reload the page, or send email to webmaster@xyz.com if the problem persists.");
```

However, if the Event Router is up when you first attempt to fetch the application, then the Microserver will be present (and fully functional), but as soon as the Router goes down, everything will stop working. In some cases you will get "permission denied" errors. Currently, the only known workaround is to send regular heartbeat events and start a watchdog timer from your onload handler:

```
// send a new pulse thirty seconds after we received the first one
watchdogInterval = 30 * 1000;
// allow up to five seconds for the pulse to make a round trip through the
// router; this is probably excessive, but slow network connections + slow
// browsers have been known to delay this long.
maxHeartbeatLatency = 5 * 1000;
// watchdog subscription URI
watchdogName = "watchdog://" + self.name;
// watchdog timer handle
bone = null;

function watchdogCallback()
{
    clearTimeout(bone);
    bone = null;
    setTimeout('watchdog()', watchdogInterval);
}

function bite()
{
    alert("We apologize, but we seem to be experiencing technical difficulties. \n" +
        "You may need to empty your browser's cache of stored web pages. \n" +
        "Please wait a bit and then reload this web page. \n\n" +
        "Please send email to webmaster@xyz.com if the problem persists.");
}

function watchdog()
{
    bone = setTimeout('bite()', maxHeartbeatLatency);
}
```


Glossary

authentication	The process of identifying an individual, usually based on a username and password. Authentication merely ensures that the user is who he or she claims to be, but says nothing about the access rights of the individual. Access rights are defined through <i>authorization</i> .
authorization	The process of granting or denying access to a network resource. Most computer security systems are based on a two-step process. The first stage is <i>authentication</i> , which ensures that a user is who he or she claims to be. The second stage is <i>authorization</i> , which allows the user access to various resources based on the user's identity.
batching	<p>Allows an HTTP client to connect to the KnowNow Event Router and send a series of requests for different operations in one HTTP request. Batches can span multiple topics. The resulting codes for each batch request are sent as part of the HTTP body.</p> <p>Always use the <code>kn_status_to</code> parameter with the JavaScript Microserver when batching. Outside the JavaScript Microserver, batching does not provide any benefits over multiple posts over a persistent connection.</p>
certificate	A certificate contains a distinguished name, public key, private key, and information about the issuer of the certificate.
Certification Authority	An entity that issues certificates, usually associated with an Authentication Server. You can check the validity of an issued certificate by checking it against the appropriate Authentication Server.
certificate file	<p>The storage location for one or more certificates. A certificate file can store certificates, trusted roots, and uncertified key-pairs. An uncertified key pair is one that has not yet been certified by the Certification Authority.</p> <p>See also, trusted roots.</p>
CLASSPATH	The environment variable that tells the Java compiler where to look for the classes it needs. <code>-classpath</code> is an option to the Java interpreter and the Java compiler that tells them (while compiling or running) where to look for a class. <code>-classpath</code> overrides CLASSPATH.

clients	A server, service, browser, device, or application.
control container	An ActiveX-enabled application or development environment that supports programmatic access to ActiveX controls.
do_max_n	Requests to the Router to deliver the last <i>n</i> events.
do_max_age	Requests to the Router to deliver all events seen in the previous <i>n</i> seconds.
duplicate event squashing	KnowNow Event Router property that allows once-only delivery of events.
encryption	The scrambling of data so that it can only be un-scrambled by an authorized recipient. Encryption strength is usually stated in bits.
event	Messages sent by an application or person specifically to trigger actions. Events have an identification string, timestamp, an expiration timestamp, metadata headers, and a content payload. Within the KnowNow event payload, events can contain data in any format. The KnowNow Event Router can handle events with <code>non-<i>kn_header</i></code> . KnowNow reserves the right to all headers that begin with <code>kn_</code> for its own use and definition.
event aggregation	Subscribing a topic to another topic. Using these inter-topic subscriptions, events can be aggregated from multiple topics to a single topic.
event forwarding	Distributed published events to all clients subscribed to a topic. The KnowNow Event Router maintains a table of subscribers for each topic so that it knows how to distribute published events. Subscriptions are automatically deleted when they expire or when the KnowNow Event Router cannot deliver an event to the given subscriber. Clients can also delete subscriptions. KnowNow Event Routers send events as HTTP POST in KN event format (which is <code>x-www.url-form-encoded</code> with special KnowNow fields), or using a JavaScript tunnel or a simple tunnel for HTTP responses. See tunnels.
event replay	A request submitted to the KnowNow Event Router to send all events that occurred since a specific point-in-time. You can specify a timestamp, request the last <i>n</i> events, or replay all stored events. This event replay request is executed immediately to only the specific subscriber that made the request.
event updating	KnowNow Event Router property that allows the contents of an event to be updated while maintaining the same event identification number.

filter	A filter can perform filtering and data transformation on events as they are routed. As events are moved into topics, out of topics, or across routes, these modules can filter, modify, and redirect the events to alternate destinations.
firewall	A system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software. The Event Router requires you to open port 8000 for non-SSL connections. For SSL connections, port 8443 is used.
JavaScript Microserver Vendor	The KnowNow Event Router can vend the JavaScript Microserver, so that the Microserver downloads into a browser automatically with a KnowNow-enabled web page. As a result, end users can use KnowNow applications that do not require additional installs to their browsers.
kn_expires	Event or route expiration time. Expressed in the same format as <code>kn_time_t</code> or in seconds relative to the event's arrival (or route creation). Shipping default is 1 hour.
kn_id	Event id. Id's are unique to each topic; if an event is published or routed to a topic which already contains an event with the same <code>kn_id</code> , the new event is considered an update. Updated events are moved to the end of the topic's event stream such that a later "route" request with <code>do_max_n</code> or <code>do_max_age</code> parameters will return the updated events.
kn_journal	A journal topic is in essence a URL that is accessible on the Internet. Microservers use the journal topic to communicate between the Router and an application that cannot otherwise receive a POST from the Router.
kn_payload	Event payload, analogous to a message body. If <code>kn_payload</code> is not present, the Router inserts this header with the empty string as its value.
kn_route_checkpoint	Only added to events when added to a <code>kn_journal</code> topic. A globally unique identifier stamped onto the event before delivery is attempted down tunnels. By supplying a checkpoint value via the <code>do_since_checkpoint</code> optional parameter in a tunnel creation request, Microservers can ask the Router to replay events they may have missed during a short disconnection.
kn_route_from	Full URI of the topic the event was last routed from. Example: <code>http://gravitar.knownow.com/kn/what/chat</code> .
kn_route_id	<code>kn_id</code> of the route the event last traversed. Example: 00192016. See <code>kn_id</code> .

kn_route_location	Full URI, including route id, of the route which the event last traversed. Example: http://gravitar.knownow.com/kn/what/chat/kn_routes/00192016.
kn_status_from	The KnowNow Event Router generates status events to indicate the success or failure of route, notify, and batch requests, as well as requests with unknown <code>do_methods</code> . Each such request, whether in a batch or by itself on an HTTP connection, generates a status event. <code>kn_status_from</code> is a value to place in the <code>kn_route_location</code> header of status events.
kn_time_t	Timestamp recording the event receipt time. Time is expressed as seconds since Jan 1 1970 00:00:00 (UNIX time ())format, e.g. "979273892" for an event created Fri Jan 12 04:31:32 2001). This header is supplied if it does not exist. The <code>kn_time_t</code> is not updated as the Router routes the event.
kn_to	Route or notify destination. e.g. /what/chat, http://foo/kn/what/chat. <code>kn_to</code> is basically used to set up routes.
kn_uri	If <code>kn_uri</code> is set as part of the HTTP header, it overrides the <code>kn_route_location</code> .
microserver	KnowNow component that connects an application to the KnowNow Event Router via HTTP. Allows application programmers to utilize the publish and subscribe features of the Router. Microservers for Solaris, Win32 (C++ and ActiveX interface), Java, and JavaScript programming environments are also available. See Developer.KnowNow.com for more information on downloading and working with these Microservers.
module	In software, a module is a part of a program. Programs are composed of one or more independently developed modules that are not combined until the program is linked.
persistent storage	Persistent storage uses operating system asynchronous I/O to write events to disk. This is used for events that should be delivered, but no significant problems will occur if they are not.
publish	The act of sending an event from a KnowNow Event Router. Any event that is sent to a topic is "pushed" to all subscribers listening on that topic. The routing of events happens instantly after the events are received. Events persist as long as the event expiration timestamp does not expire. See topic and <code>kn_expires</code> .
route	The path an event travels during the publish/subscribe process.

secure sockets	A socket is a way of sending information over a network between two programs. Secure sockets make this communication safe. Data is encrypted so only people who the data is meant for can read it. See <i>also</i> encryption.
SOAP	Simple Object Access Protocol. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules, and a convention for representing remote procedure calls and responses.
SSL	Secure Socket Layer standard. A protocol that delivers server authentication, data encryption, and message integrity. SSL is layered beneath application protocols such as HTTP, SMTP, Telnet, FTP, Gopher, and NNTP, and layered above the connection protocol TCP/IP. This strategy allows SSL to operate independently of the Internet application protocols. With SSL implemented on both the client and server, your Internet communications are transmitted in encrypted form. Information you send can be trusted to arrive privately and unaltered to the server you specify. See <i>secure sockets</i> .
subscribe	The act of receiving an event from the KnowNow Event Router. When a client connects to a KN Event Router and subscribes to a topic, the client begins to receive events that are published to that topic. The client continues to receive events, until the subscription has expired. See <i>topic</i> .
timestamp	Timestamp recording the original event receipt time. Time is expressed as seconds since Jan 1 1970, 00:00:00 (UNIX time()) format, e.g. "979273892" for an event created Fri Jan 12 04:31:32 2001).
topic	<p>A collection of events to which KnowNow clients can subscribe or publish events. All events are published to a topic. A topic can be thought of as a database table where events are stored. As updates occur, all subscribers are notified of the changes. Subscribers also have the option of selecting a sub-set of the data. Duplicate events that are published to a topic are automatically deleted by the KnowNow Events Router. You can attach metadata to a topic by using the <code>kn_metadata</code> header. The KnowNow Event Router has a configurable payload size threshold to discard large events without overloading.</p> <p>Clients can use a Uniform Resource Locator (URL) to refer to and access topics on the KnowNow Event Router. Topics are arranged in a naming hierarchy that uses forward slashes (/) to separate the levels. For example: <code>/what/nasdaq/SUNW/bid</code>.</p>
topic management	Allows the user to create, edit, or delete topics and subtopics.

trusted root	A special certificate issued by a well-known and trusted Certification Authority.
tunnel	The physical route that connects a KnowNow Event Router to a client, like a Microserver or a standalone HTTP application. Tunnels use persistent connections to send events to desktops and applications while maintaining the policies of Internet administrators for their firewalls, Network Address Translators (NATs), and Web proxies. To maintain a persistent connection, tunnels occasionally send <i>keepalives</i> to the KnowNow Event Router. Keepalives are single byte events. A client can request that the KnowNow Event Router close a tunnel at a specific time by setting the <code>kn_expires</code> header parameter.
URI	Uniform Resource Identifier.