RFTagAware™

Deployment Guide

Version 1.3

June 2005

ConnecTerra, Inc.

100 CambridgePark Drive

Cambridge, Massachusetts 02140

Phone: (617) 441-2200

Facsimile: (617) 492-5837

Web Site: www.connecterra.com

Email: info@connecterra.com

ConnecTerra® is a leading provider of enterprise software for device computing.

Contents

Preface

Purpose of This Manual

This manual describes how to install, configure, and use RFTagAware software.

Audience

- System administrators and other IT professionals responsible for installing and deploying RFTagAware.
- Software engineers and other developers using the Application Level Events (ALE) API.
- IT procurement specialists.
- Network architects and engineers.
- Technology strategists and senior managers.

Users should have knowledge of systems, networking, and installing applications on Microsoft® Windows and UNIX® systems.

Related Documents

- *RFTagAware Reader Configuration Guide*
- *RFTagAware Programmer Guide*

What's in This Manual

- Chapter 1: Introduction

  This chapter provides an overview of the RFTagAware product.
- Chapter 2: Installing and Configuring RFTagAware

  This chapter describes how to install and configure RFTagAware.
- Chapter 3: Using the Administration Console

  This chapter describes how to use the Administration Console.
- Chapter 4: Using the Reader Simulator

  This chapter describes how to use the RFID Reader Simulator included with RFTagAware.

Contacting Technical Support

For technical support Monday-Friday 9am-5:30pm Eastern Time, call 617-441-2200, or send email to support@connecterra.com.

ConnecTerra, Inc.
100 CambridgePark Drive
Cambridge MA 02140
Voice: +1-617-441-2200
FAX: +1-617-492-5837

Chapter 1: Introduction

Contents

This chapter provides an overview of RFTagAware™.

RFTagAware Overview

ConnecTerra RFTagAware software provides comprehensive software infrastructure for developing and deploying Radio Frequency Identification (RFID) applications. RFTagAware is designed to solve two problems that arise in any enterprise-scale deployment of RFID technology:

- Data Filtering and Integration

  RFID readers generate a flood of very fine grained information. For example, an RFID reader deployed in a typical warehouse shelf environment will report the presence of a particular tag as frequently as five times per second. Most business applications cannot deal effectively with such fine grained information; they would rather receive reports such as "in the last 60 seconds, the following case-level tags were added to the shelf."

  RFTagAware provides a simple yet powerful application programming interface (API) through which applications can define high level events such as this, and receive them in a variety of formats designed for easy integration with enterprise software. Applications define what application-level events they are interested in, and RFTagAware fulfills those requests by interacting with RFID readers and performing filtering, counting, and grouping operations.

- Management of RFID Infrastructure

  Enterprise-scale RFID applications typically involve hundreds to hundreds of thousands of RFID readers, deployed at dozens to thousands of remote sites. Most readers are deployed at sites having limited or no IT support staff. Moreover, RFID readers operate autonomously, generating data into enterprise business processes without human intervention or control. In order for business processes to rely on this data, there must be confidence that the infrastructure is sound. For example, it makes a difference whether no RFID tags are read at Loading Dock #5 because no shipments are received, or because an errant forklift damaged the reader's antenna.

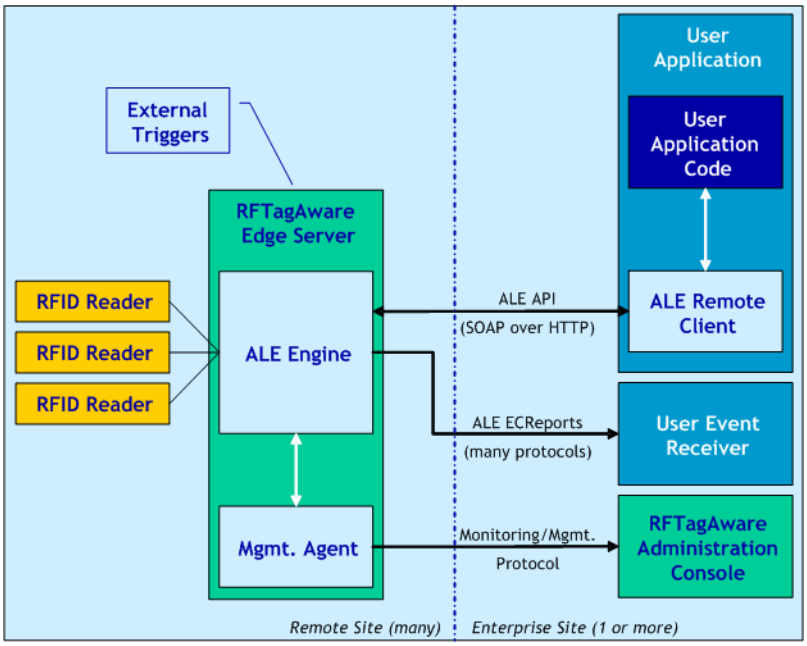
  RFTagAware provides a comprehensive remote monitoring and management facility for RFID infrastructure. Through RFTagAware's monitoring and management facilities, operations staff can centrally monitor the health and functioning of remote RFID readers and associated computer hardware, and also perform remote maintenance operations such as upgrading software, running diagnostics, and suspending or resuming operation of any component. RFTagAware can also connect this information into existing enterprise infrastructure management and service applications.

These two features together provide a comprehensive platform for the development of reliable, enterprise-scale RFID applications. With RFTagAware, applications can be written at a suitably high level, while RFTagAware takes care of all the low-level interactions with readers of many types, and makes sure the network of readers and software always functions at peak efficiency.

RFTagAware Architecture

RFTagAware software has two main components:

- Edge Server (page 1-3)
- Administration Console (page 1-3).



Edge Server

The **Edge Server** is software deployed at a remote site where there are RFID readers.

There are two main subcomponents within the Edge Server:

- The first is an Application Level Events (ALE) processing engine. The ALE Engine is responsible for receiving and processing RFID tag data on behalf of user applications.
- The second is a monitoring and management agent that allows the health and functioning of readers and edge server software to be monitored and managed remotely through the RFTagAware Administration Console.

Administration Console

The Administration Console is software typically deployed at a centralized enterprise site, where operations staff may use it to configure and monitor the activities of multiple Edge Servers and readers. The Administration Console can:

- Monitor the activities of multiple Edge Servers and readers.

- Add, remove, and configure readers and composite readers
- View and edit a list of `ECSpec` data objects and their subscribers.

For detailed information on the Administration Console, see Chapter 3: Using the Administration Console.

Application Programming Interface (API)

Applications interact with the Edge Server through an application programming interface (API) called Application Level Events, or ALE. The ALE interface provides a high-level, declarative way for applications obtain RFID data, without requiring application programmers to interact directly with RFID readers or perform any low-level real-time processing or scheduling operations.

For detailed information about the ALE interface, see the *RFTagAware Programmer Guide*.

Standards Compliance

RFTagAware is compliant with all relevant standards of EPCglobal. ConnecTerra is committed to revising RFTagAware so that it is always in compliance with these standards.

RFTagAware conforms to EPCglobal standards in the following way:

- RFTagAware works with RFID readers that implement the following protocols:
  - EPCglobal Class 0, Class 0+, and Class 1 RF Protocols
  - ISO 15693 RF Interface protocol
  - ISO 18000-6B RF Interface protocol

  RFTagAware will also work with RFID readers that implement the forthcoming EPCglobal UHF Generation 2 RF protocols, when such readers become available.
- RFTagAware supports the EPCglobal Tag Data Standard. This standard governs the bit-level encoding of object identity and other information onto RFID tags. It also specifies a URI-based syntax for exchange of tag data between software application components, and a second URI-based syntax for the description of filtering patterns. RFTagAware fully implements all of these standards.
- RFTagAware includes components that fulfill the collecting and filtering role, as defined by EPCglobal. RFTagAware also provides other value add functionality in the areas of reader management and application integration. RFTagAware supports EPCglobal standards by providing a standard API compliant with the ALE specification.

- The Application Level Events API was developed by ConnecTerra and other EPCglobal member companies under the auspices of the EPCglobal Software Action Group. ALE has been standardized by EPCglobal. ConnecTerra is committed to the continued evolution of ALE as a standard.

What's Next: Quick Start Overview

To get started using RFTagAware, take a look at the Quick Start Overview (page 2-2).

Chapter 2: Installing and Configuring RFTagAware

Contents

This chapter describes how to install and configure RFTagAware.

Quick Start Overview

Here is a quick list of tasks that will get you started with RFTagAware:

- Install the RFTagAware software.

  See Installing RFTagAware on page 2-3.

- Set up your development environment.

  See the *RFTagAware Programmer Guide*.

- Run the sample applications, and take a look at the descriptions of the sample source code.

  See the *RFTagAware Programmer Guide*.

System Requirements

RFTagAware is supported on the following platforms:

- Microsoft Windows 2000 Professional, Windows 2000 Server, Windows XP, and Windows 2003 Server.
- Intel Pentium-compatible running Red Hat Linux Version 8 or Version 9, or Fedora Core 3.
- Sun Sparc running Solaris 8 or Solaris 9.

A full stand-alone installation of RFTagAware requires approximately 60 MB (UNIX) or 40 MB (Windows) of disk space, excluding prerequisite software.

Prerequisite Software

RFTagAware does not require any other software to be pre-installed. The following software is optional:

- RFTagAware is implemented in Java™ and includes a bundled Java Runtime Environment (JRE). During installation on Windows platforms, you may optionally choose to use a different JRE or Java Development Kit (JDK™) that you have previously installed, provided that it is compatible with JDK 1.4 or later. If you select this option during installation, you will be prompted for the location where you previously installed your JRE or JDK.
- If you want to modify and recompile the Sample Applications provided with RFTagAware, you must install a Java Development Kit 1.4 or later.

  You can download the JDK from:
  `http://java.sun.com`

Supported RFID Readers

You need one or more RFID readers to use RFTagAware. A list of supported readers appears in the *RFTagAware Reader Configuration Guide*. ConnecTerra is committed to providing support for new makes and models of RFID readers as they become available, so check with your ConnecTerra sales representative if your selected RFID reader does not appear in the list of supported readers.

Some readers may require specific configuration prior to use with RFTagAware. The *RFTagAware Reader Configuration Guide* also provides configuration details for supported readers.

If you do not have an RFID reader, you can use the reader simulator provided with RFTagAware. The reader simulator runs on any workstation. Out of the box, it provides a minimal simulation of a ThingMagic Mercury4 reader; with minor configuration it can provide a minimal simulation of a Printronix printer. The reader simulator is useful for software evaluation, application development, and debugging.

Installing RFTagAware

This section shows you how to install RFTagAware.

If you are planning to modify and recompile the Sample Applications, make sure you have installed the Java Development Kit, version 1.4 or later.

Installation Modes

There are three basic installation modes:

An additional fourth mode is available if you have already installed this version of RFTagAware, and then reinstall this same version again:

Full Installation Mode

The **Full** install places all the components on a single machine, sets the configuration parameters accordingly and provides documentation and sample code demonstrating the use of the Java API and `ECSpec` and `PCSpec` syntax. This provides a good initial development and exploration environment for the RFTagAware product.

Custom Installation Mode

The **Custom** install allows you to place individual components on a particular machine, or to omit certain components entirely. You would choose the custom install to, for example, place a single Administration Console on an administration or monitoring machine, Samples and Documentation on developer machines and Edge Servers on other remote machines.

The Custom install requires a certain familiarity with the product, and in some cases will require you to make changes to property or configuration files once the installation is complete.

Upgrade Installation Mode

The **Upgrade** install lets you upgrade from a previous version of RFTagAware. Upgrade assumes you have a 1.0 or later version of RFTagAware already installed in your environment. Upgrade lets you install the latest software while preserving your existing configuration information.

An upgrade never occurs on top of an existing installation. Your previous installation is always preserved unmodified.

Note that you can upgrade **only** from 1.0 or later versions of RFTagAware. If you have an earlier version (0.9.x), you need to do a Full or Custom install.

Add Features/Reinstall Installation Mode

The **Add Features/Reinstall** mode appears only when you run the installer in an environment where you already installed **this version** of RFTagAware. It lets you:

- Add additional components to the current version of RFTagAware.

  You might want to do this if you initially installed a subset of components, then later want to add additional components.

- Reinstall keeping your configuration information.
- Reinstall overwriting your configuration information.

Directory Structure Concepts

Defaults and Allowed Install Locations

RFTagAware's default installation directories are:

- Windows: `C:\Program Files\ConnecTerra\RFTagAware`
- UNIX: `/opt/ConnecTerra/RFTagAware`

You can choose different primary install locations.

Directory Structure

Regardless of where you put your primary install directory, RFTagAware creates two main subdirectories beneath the primary install directory:

- `control`

  Version-independent location for the master startup scripts for RFTagAware's various components. These scripts act like pointers or symbolic links to the actual scripts for a particular version. This lets you continue to use the same `/control/bin/script_name` path to run a script, no matter how many times you upgrade.

- `version_number`, for example:
  `1.3`

  Contains the files for a specific version of RFTagAware.

Directory Tree Overview

Here are some highlights from a sample installed directory tree:

`control`	

`control/bin`	Use these master scripts to start the Administration Console, Edge Server, Reader Simulator, and Quick Test utility:
	`RunAdminConsole`
	`RunEdgeServer`
	`RunReaderSim`
	`RunQuickTest`
	These scripts will always invoke the version of RFTagAware that you have most recently installed.

1.3	
1.3/bin	Version-specific scripts used by RFTagAware. Use these scripts to start a *particular* version of RFTagAware, even if it is not the version you most recently installed.
1.3/etc	Properties and logging files for the Administration Console, Edge Server, and JMS Notification Driver: admin-console.props edge.props jms.options logging.props.admin-console logging.props.edge
1.3/lib	Java libraries used by RFTagAware components.
1.3/samples	Sample Java source code that illustrates how to program the ALE API.
1.3/share/schemas	Schema used to represent RFTagAware data types in XML and WSDL files describing the ALE API.
1.3/UninstallerData	Files used by the installer during uninstallation.
1.3/var	Files created and used by RFTagAware during operation, including log files. The edgestate subdirectory contains state data about ECSpec, PCSpec, and EPCCacheSpec instances and their subscribers, and reader configuration data for readers configured using the Administration Console, as well as other persistent data that you create using the ALE API.

Note: The version number (1.3) shown above will be different depending on the specific version of RFTagAware installed on your system.

Shortened Pathnames in Documentation

RFTagAware documentation refers to the RFTagAware subdirectories (for example, `bin` and `etc`) directly, without the preceding full pathname. In all cases, these shortened pathnames refer to subdirectories within the default or user-specified RFTagAware installation directory.

Descriptions of All Installation Screens and Fields

The tables below contain detailed descriptions of each check box and input field on every installation screen.

- Depending on the installation mode and components you choose, you may not see all of these fields during your installation. Simply ignore the field descriptions that do not apply to you.
- You might want to assemble the information that pertains to you before you run the installer. (Instructions for running the installer, along with sample screenshots are in Procedures for Running the Installer: A Sample Full Installation on page 2-11).
- The install program will ask you to fill in (or accept default values for) most of the following fields. Where there are defaults, you can generally accept them.

Screen Name: Choose Install Folder	
Where Would You Like to Install?	The path to the primary directory where you want to install RFTagAware. Suggested defaults: Windows: `C:\Program Files\ConnecTerra\RFTagAware` UNIX: `/opt/ConnecTerra/RFTagAware` • You can choose a different primary install location. • If you are running the installer in an environment where you already installed **this version** of RFTagAware, be sure to specify the primary directory for the current RFTagAware installation.

Screen Name: Choose Deployment Type	
Full	A Full installation installs all components. Choose **Full** if you want to install all components on the same machine.
Custom	RFTagAware has several install components: • Edge Server – Interacts with RFID readers, and delivers tag data to applications via the ALE API. Managed by the Administration Console. • Administration Console – Provides remote administration and monitoring of Edge Servers and their readers. • Reader Simulator – Standalone application that simulates a ThingMagic Mercury4 reader with two UHF antennas. • Sample Code – Java source code that illustrates how to program to the ALE API. • ALE API Documentation – Provides the *RFTagAware Programmer Guide*, the *RFTagAware Deployment Guide* (this document), the *RFTagAware Reader Configuration Guide*, and the ALE API Java Documentation. Choose **Custom** in any of the following situations: • You do not want to install all components. • You want to install components on different machines.
Upgrade	Choose **Upgrade** to upgrade from a previous version of RFTagAware.
Add Features/Reinstall	The **Add Features/Reinstall** mode appears only when you run the installer on a machine with **this version** of RFTagAware is already installed. It lets you: • Add additional components to the current version of RFTagAware. You might want to do this if you initially installed a subset of components, then later want to add additional components. • Reinstall keeping your configuration information. • Reinstall overwriting your configuration information, and using the default settings instead.

Screen Name: Upgrade an Existing Configuration	
Please choose a folder	This screen appears only if you chose the **Upgrade** deployment type. Click **Choose** and browse to the primary installation directory for the previous version of RFTagAware (version 1.2, or 1.2.x), for example: `C:\Program Files\RFTagAware\1.2.1`

Screen Name: Components This screen appears only if you chose the **Custom** or **Upgrade** deployment type.	
	This screen lets you choose which of the following components you want to install: • Edge Server • Administration Console • Reader Simulator • Sample Code • ALE API Documentation Check the components you want to install. Uncheck the components you do not want to install. If you are doing an upgrade, you need to check all the components you want to upgrade. If you do not check a component, the installer does not upgrade it.

Screen Name: Choose Java Virtual Machine This screen appears only if you chose the **Custom** deployment type.	
RFTagAware requires a Java Virtual Machine (VM). You can use the VM that is part of a Java Runtime Environment (JRE) or the one that is part of a Java Software Development Kit (JDK). **Important Requirement for Sample Applications:** If you plan to modify and recompile the RFTagAware sample Java applications, you need to specify a JDK (version 1.4 or higher) here.	
Install a Java VM specifically for this application	Choose this option if you want to use the Java VM (JRE) that comes bundled with RFTagAware.
Choose a Java VM already installed on this system.	Choose this option if you want to use a previously installed JRE or JDK. Java VMs are only detected on Windows; for Unix installations, this box is blank.

Screen Name: Edge Server Configuration (pt 1) Site ID and Edge Server ID are logical names that you devise to describe various locations in your enterprise. The Administration Console uses these names to report on reader activity at the various locations. For more information and examples of how these names are used in practice, see Determining Your Site IDs and Edge Server IDs on page 2-10.	
Site ID	Site IDs typically represent building names, for example: `Warehouse4`
Edge Server ID	Edge Server IDs typically represent smaller locations where an individual Edge Server is monitoring readers, for example: `LoadingDock2`

Screen Name: Edge Server Configuration (pt 2)	
Edge Server ALE Service Port	This is the port **on the Edge Server machine** that your applications use to access the ALE API. Choose any port that is not being used by another application.

Screen Name: Edge Server Configuration (pt 3)	
Monitoring Rate (ms)	How frequently the Edge Server sends telemetry information to the Administration Console. Expressed in milliseconds. Default: 1000

Screen Name: Administration Console Configuration (pt 1)	
Administration Console Telemetry Port	The Edge Server needs to report reader activity (telemetry data) to the Administration Console. This field is the port on the Administration Console that the Edge Server should use to report this telemetry information.

Screen Name: Note - Identifying a Reader

This note appears ONLY if you:

- Chose a Custom/Upgrade deployment, and
- Did not choose to install the Reader Simulator.

If these two conditions apply, RFTagAware assumes you are using a real reader, and need to configure it using `edge.props` or the Administration Console.

See Setting Up Readers on page 2-21 and the *Reader Configuration Guide* for information on settings for supported readers.

Screen Name: Note - Reader Simulator

This screen appears if you install the Reader Simulator without the Edge Server – for example, if the Edge Server and the Reader Simulator are on two different machines. In this case, you need to configure the Reader Simulator using `edge.props` or the Administration Console on the Edge Server machine.

Replace the sample values below with appropriate information for your Reader Simulator location.

```
com.connecterra.ale.reader.SimReadr.class =
com.connecterra.ale.readertypes.ThingMagicMercury4PhysicalReader
com.connecterra.ale.reader.SimReadr.hostname = edgeserver.company.com
com.connecterra.ale.reader.SimReadr.port = 5050
com.connecterra.ale.reader.SimReadr.defaultRate = 350
com.connecterra.ale.reader.SimReadr.uhf1LogicalReaderName = ConnecTerra1
com.connecterra.ale.reader.SimReadr.uhf2LogicalReaderName = ConnecTerra2
```

See Setting Up Readers on page 2-21 and the *Reader Configuration Guide* for information on settings for supported readers.

Screen Name: Sample Code Configuration	
This screen appears when the Sample Code component is installed without the Edge Server. The ALE API sample code requires a running Edge Server in order to run. You will need to specify the Edge Server hostname and port.	
Edge Server ALE Hostname	This field appears only when you are installing the sample applications in a Custom or Upgrade deployment type. Because the Custom/Upgrade deployment assumes you are may be installing components on different machines, specify the host name of the Edge Server.
Edge Server ALE Service Port	This is the port on the Edge Server (whose hostname you specified above) that the sample applications should use to access the ALE API. Choose any port that is not being used by another application.

Screen Name: Note - JDK Required to Compile Samples

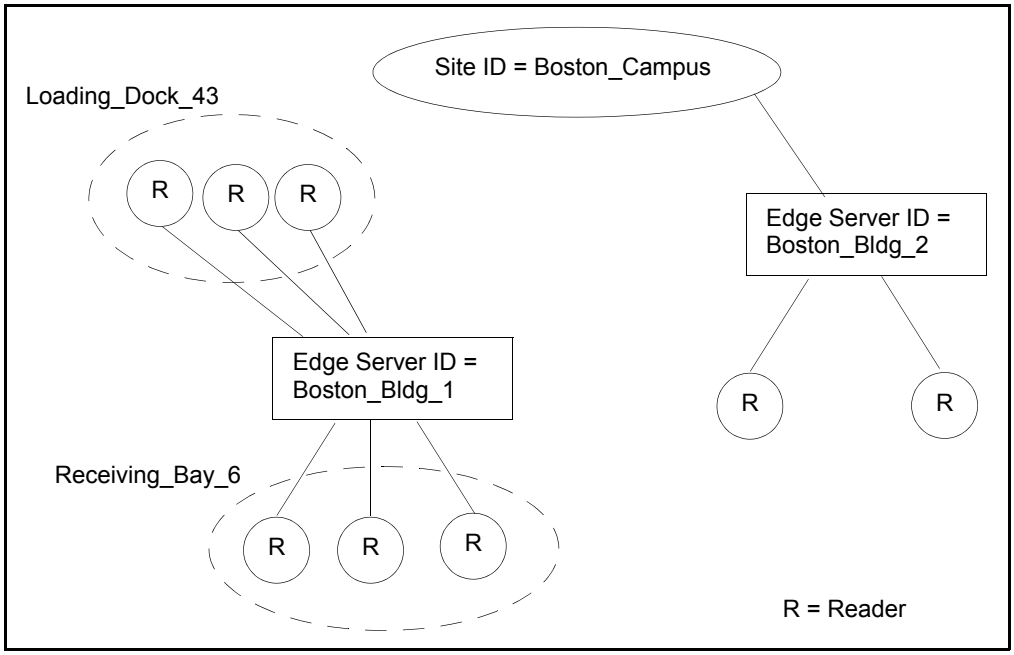
This note appears only if you are installing the sample applications AND specified a JRE (rather than a JDK) for your Java VM.
The sample applications' build scripts need the location of a JDK, in order to compile the Java source code. You can either:
• Rerun the installer, choose Reinstall (new configuration), and specify the location of a JDK, or
• Edit the sample applications' build files manually, as described below.
This example shows how to edit the build file for `ImmediateSample.java`.
• Open the file `/samples/ImmediateSample/build.bat`
• Note that its `JDK_HOME` environment variable is empty:
`set JDK_HOME=`
• Edit the `JDK_HOME` variable to point to the location of an installed JDK, for example:
`set JDK_HOME =C:\j2sdk1.4.2_01`
Edit the corresponding build files for all the other sample applications.

Determining Your Site IDs and Edge Server IDs

Site IDs and Edge Server IDs are logical names that you create to identify various locations in your enterprise. The Administration Console uses these names to report on the reader activity that is going on in each location.

A Site ID typically refers to a geographic location, such as a building or a campus. An Edge Server ID names a particular instance of an Edge Server. In many cases, one Site will have only a single Edge Server, responsible for all of the readers within that site. When a Site refers to a very large geographic area, such as a campus, there may be more than one Edge Server; for example one Edge Server in each building within the campus.

This diagram shows you a sample Site ID and its associated Edge Server IDs and readers. Note that spaces are NOT ALLOWED in Site ID and Edge Server ID names.



Procedures for Running the Installer: A Sample Full Installation

Prerequisites

If you are installing on Linux or Solaris, you must use a graphical environment to run the installer. Specifically, you must have a valid X Window System `DISPLAY` variable set in your environment.

In addition, if you are installing onto a Fedora Core 3 machine, you must add the `xorg-x11-deprecated-libs` package before starting the installer.

Full Deployment Sample

This section provides screenshots and descriptions of a sample installation, where the user chooses a **Full** deployment type. Other deployment types will differ.

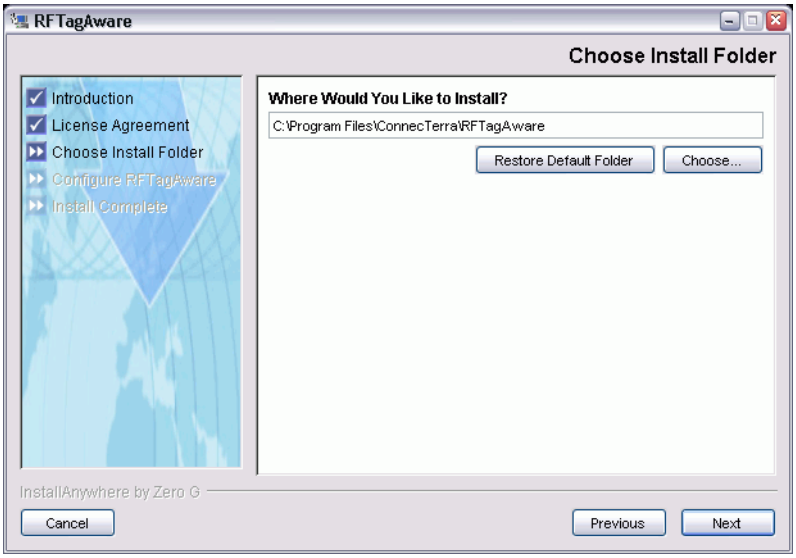
For information on the screens and fields you see if you choose other deployment types (**Custom**, **Upgrade**, **Add Features** or **Reinstall**), see Descriptions of All Installation Screens and Fields on page 2-6.

Running the Installer

1. Insert the RFTagAware CD into the machine where you want to install, and run the install program for your platform:
   (Windows) `win_rftagaware_install_1.3.exe`
   (Linux) `lin_rftagaware_install_1.3.bin`
   (Solaris) `sol_rftagaware_install_1.3.bin`

2. The installer walks you through the introduction screen and licence agreement.

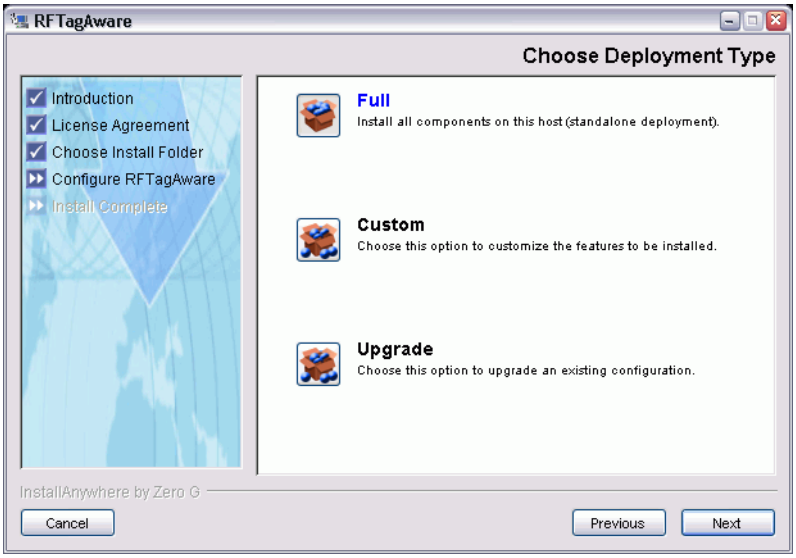
   The Choose Install Folder screen appears.



3. Click the **Choose** button to browse to the directory where you want to install RFTagAware.

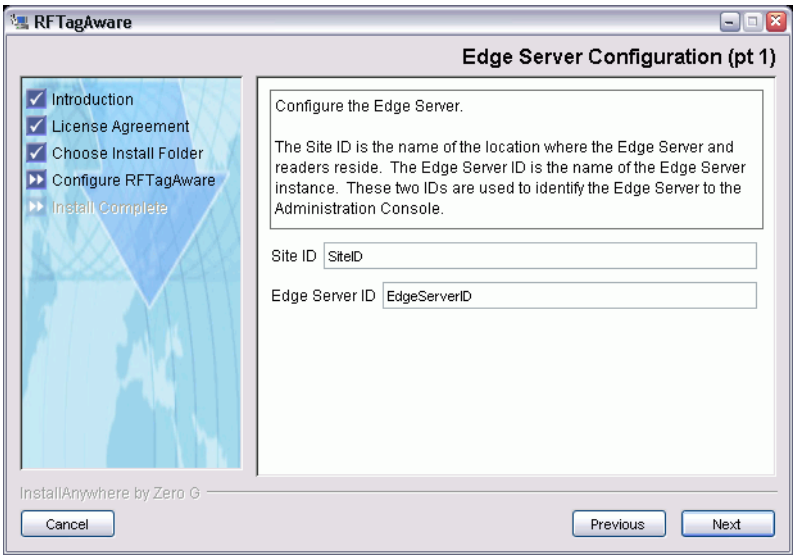
   Select your chosen directory and click **Next**.

   The Choose Deployment Type screen appears.



4. In this example, choose **Full** to install all components on the same machine, and click **Next**. This section covers the case where you choose **Full**.

The Edge Server Configuration (pt 1) screen appears.

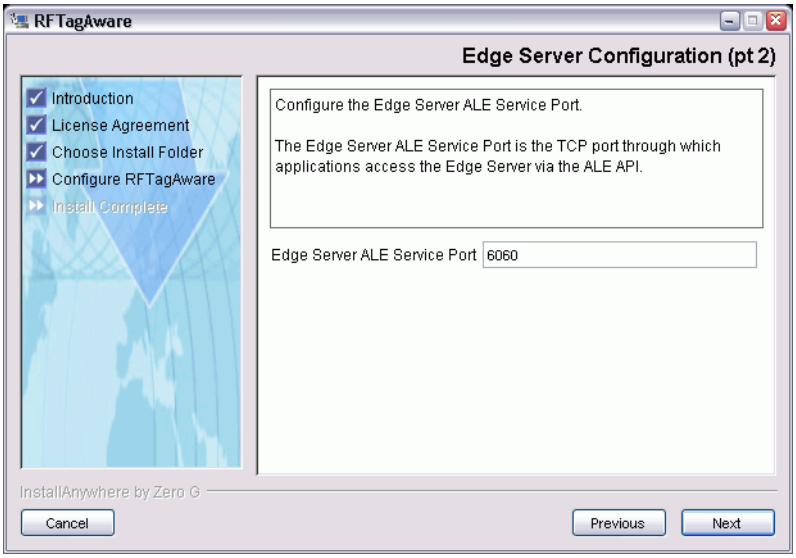


Site ID and Edge Server ID are logical names that you devise to describe various locations in your enterprise. The Administration Console uses these names to report on reader activity at the various locations. For more information and examples of how these names are used in practice, see Determining Your Site IDs and Edge Server IDs on page 2-10.

- **Site ID** – Site IDs typically point to building names, for example: `Warehouse4`
- **Edge Server ID** – Edge Servers typically point to smaller locations where an individual Edge Server is monitoring readers, for example: `LoadingDock2`

5. Type a Site ID and Edge Server ID for this Edge Server, and click **Next**.

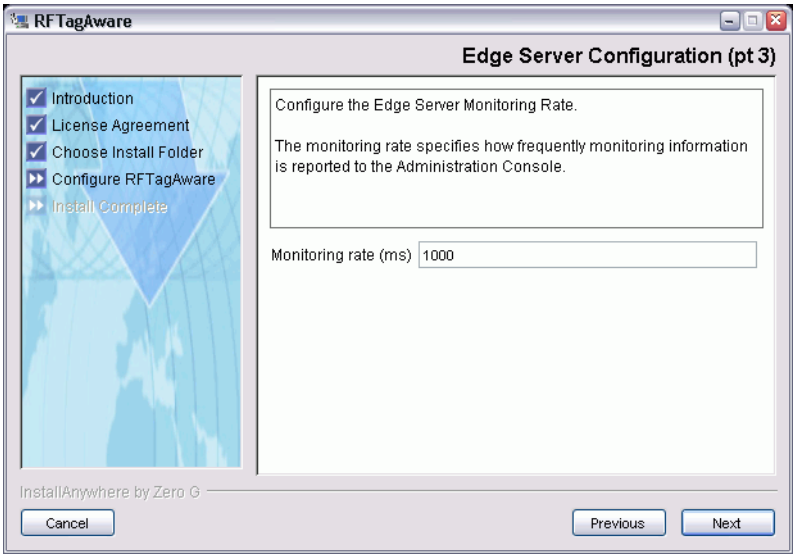
   The Edge Server Configuration (pt 2) screen appears.



The Edge Server ALE Service Port is the port **on the Edge Server machine** that your applications use to access the ALE API.

6. Type a port number that is not being used by another application, and click **Next**.

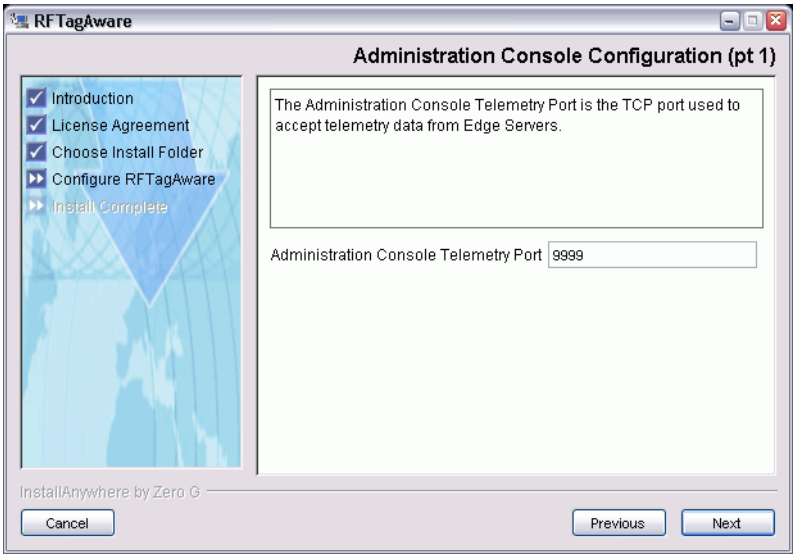
   The Edge Server Configuration (pt 3) screen appears.



   The Monitoring Rate (ms) controls how frequently the Edge Server sends telemetry information to the Administration Console. Expressed in milliseconds; defaults to 1000. If this value is less than 1, no telemetry will be sent.

7. Type a monitoring rate value and click **Next**.

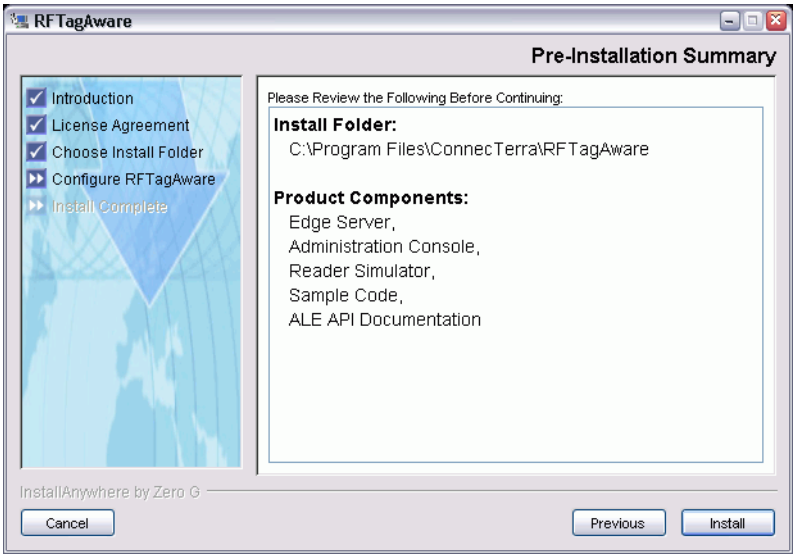
   The Administration Console Configuration (pt 1) screen appears.



   The Administration Console Telemetry Port is the port **on the Administration Console machine** that the Edge Server should use to report reader activity (telemetry information) to the Administration Console.

8. Type the Administration Console port that the Edge Server should use, and click **Next**.

   The Pre-Installation summary screen appears.



9. Click **Install** to complete the installation procedure.

Testing the Installation

You can test that RFTagAware is correctly installed and running as soon as your installation is complete by:

- Starting the simulator (`bin\RunReaderSim.bat` or `bin/RunReaderSim.sh`).
- Starting the Edge Server (`bin\RunEdgeServer.bat` or `bin/RunEdgeServer.sh`).
- Running the QuickTest application (`bin\RunQuickTest.bat` or `bin/RunQuickTest.sh`).

The simulator is preconfigured to see seven GID-64-I tags in each of two antennas, and the QuickTest application will report that it sees these tags. (For more information on using the simulator for testing, see Chapter 4: Using the Reader Simulator.) The simulator's properties definitions in `edge.props` are examples of the properties definitions required to configure RFTagAware to communicate with any reader. See the *RFTagAware Reader Configuration Guide* for the defined properties and values for each of the devices supported by RFTagAware.

Configuring RFTagAware

RFTagAware may be configured by editing the edge.props file. This file is a configuration properties file for the Edge Server. It is located in:

```
RFTagAware_install_dir/RFTagAware_version/etc/edge.props
```

where `RFTagAware_install_dir` is the directory where you installed the RFTagAware software and `RFTagAware_version` is the directory where the latest version of RFTagAware is installed on your system.

This file is formatted as a Java properties file, where each line defines a configuration parameter. Lines beginning with a pound sign (#) character are comments and are ignored by RFTagAware.

The installer writes initial configuration settings to this file. You may need to edit `edge.props` with a text editor to change your configuration.

Some setup tasks you might want to do include:

- Setting Up Readers (page 2-21)
- Specifying State Data Persistence (page 2-26)
- Configuring the HTTP Notification Driver (page 2-28)
- Enabling the SNMP Log Handler (page 2-28)
- Setting up the JMS Notification Driver (page 2-29)

This section contains a sample version of `edge.props` (shown below). Contact your ConnecTerra representative for more detailed information on editing this file.

Sample edge.props File

```
# Copyright (C) 2005, 2003-2004 by ConnecTerra, Inc.
# All rights reserved; use is subject to license terms.
# US and international patents pending.

# ================================================================
#
# Edge Server system properties
#
# These properties configure the general behavior of the edge server.
#

# The TCP port on which the ALE service listens.
com.connecterra.ale.servicePort = 6060

# The identity for this Edge Server. This name appears as the savantID in ECReports,
# and also as the device ID for telemetry associated with the Savant as a whole.
com.connecterra.ale.savantID = EdgeServerID

# The site identifier for this Edge Server
com.connecterra.ale.siteID = SiteID

# The maximum number of threads that will simultaneously attend to event cycles
com.connecterra.ale.eventCycleConcurrency = 10

# The maximum number of events which can be pending against a single event cycle. This
# defaults to -1, which means no limit. If an event cycle gets too busy based on this
# parameter, read cycles may be dropped and some API calls may block.
```

```
com.connecterra.ale.eventCycleEventQueueSize = 200

# The maximum number of threads that will simultaneously attend to programming cycles
com.connecterra.ale.programmingCycleConcurrency = 10

# The maximum number of events which can be pending against a single event cycle. This
# defaults to -1, which means no limit. If an event cycle gets too busy based on this
# parameter, read cycles may be dropped and some API calls may block.
com.connecterra.ale.programmingCycleEventQueueSize = 200

# A space separated list of URLs, each specifying a location of the EPC Company Prefix
# Index translation table. This table is necessary to decode tags encoded in the
# SGTIN-64, SSCC-64, and SGLN-64 formats as defined by the EPCglobal Tag Data
# Specification Version 1.1. At initialization time, the edge server attempts to load
# the translation table from each URL, from left to right, stopping after the first
# successful load. Typically, the first URL in the list is the public HTTP URL
# provided by EPCglobal, and the second is a file: URL naming a local file. The local
# file would then be used as a backup in case the EPCglobal web site is not accessible.
#
# For example:
#
# com.connecterra.ale.epcIndexTableURL = http://www.onsepc.com/ManagerTranslation.xml
file:ManagerTranslation.xml
#
# The contents of the translation table file is an XML document having the following
# form:
#
# <GEPC64Table>
#  <entry index="1" companyPrefix="0037000"/>
#  <entry index="2" companyPrefix="0041333"/>
#  ...
#  <entry index="16383" companyPrefix="0614141"/>
# </GEPC64Table>
#
com.connecterra.ale.epcIndexTableURL = http://www.onsepc.com/ManagerTranslation.xml
file:ManagerTranslation.xml

# ================================================================
#
# Persistence properties
#
# These properties specify how and when persistence should be used.
#

# Persistence is disabled if set to true, and all the other properties in this section
# are ignored. Defaults to false.
com.connecterra.ale.persistence.disabled = false

# If false, then any inconsistency in the persisted data will cause the engine not to
# start. If true, the persistence manager will use as much data as it can, and move the
# bad data aside. Defaults to false.
com.connecterra.ale.persistence.allowBadData = false

# The directory in which the persistence data is stored. This property is required if
# disabled is false. No default value.
com.connecterra.ale.persistence.directory = ../var/edgestate

# Each time this interval (in milliseconds) passes, all persistence data which has not
# yet been written out is written out.  Any value less than or equal to zero means
```

```
# never. Defaults to never.
com.connecterra.ale.persistence.interval = 0

# If this is true, then event cycle data is written out every time the cycle is
# activated.  Defaults to false.
com.connecterra.ale.persistence.persistOnActivate = false

# ================================================================
#
# Dynamic Configuration properties
#
# These properties configure the dynamic configuration mechanism.
#
com.connecterra.ale.dynamicConfig.enabled = true

# ================================================================
#
# Telemetry properties
#
# These properties configure the telemetry data gathering and delivery mechanisms.
#

# The period (in milliseconds) for sending telemetry If unspecified or non-positive, no
# telemetry will be sent
com.connecterra.ale.telemetryPeriod = 1000

# ================================================================
#
# Notification delivery failure properties
#
# These properties describe the system-wide behavior when notification delivery fails.
#

# For subscriptions where no failure count is specified, this is used.
com.connecterra.ale.notifyDefaultFailureLimitCount = 0

# For subscriptions where no failure interval is specified, this is used. Units are
# milliseconds.
com.connecterra.ale.notifyDefaultFailureLimitInterval = 0

# This property provides an overriding maximum failure count for all subscriptions. If
# a user requests a count greater than this value, this value is used instead.
com.connecterra.ale.notifyMaximumFailureLimitCount = 0

# This property provides an overriding maximum failure interval for all subscriptions.
# If a user requests an interval greater than this value, this value is used instead.
# Units are milliseconds.
com.connecterra.ale.notifyMaximumFailureLimitInterval = 0

# The maximum number of threads that will simultaneously attend to delivery of
# notifications, the default value is 3.
com.connecterra.ale.notifyConcurrency = 10

# ================================================================
#
# Notification driver properties
#
# These properties configure the notification drivers
#
```

```
# For each notification driver to be defined, include a set of properties like this:
#
# com.connecterra.ale.notificationDriver.<URIScheme>.class = <className>
# com.connecterra.ale.notificationDriver.<URIScheme>.<prop1> = <prop1value>
# com.connecterra.ale.notificationDriver.<URIScheme>.<prop2> = <prop2value>
# ...
# where prop1, prop2, etc, are defined by the driver class.
# URIScheme is the URI scheme to which the specified driver applies.

# URIs of the form "console:<headingComment>"; e.g., "console:testDisplay"
# The notification is written to the console in indented XML, with a comment that
# includes the headingComment string.
com.connecterra.ale.notificationDriver.console.class =
com.connecterra.ale.notifytypes.ConsoleNotificationDriver

# URIs of the form "http:..." interpreted as an HTTP URL;
# e.g. "http://connecterra.com:8080/bar/quux"
# The notification is sent in XML to the specified URL using HTTP POST.
com.connecterra.ale.notificationDriver.http.class =
com.connecterra.ale.notifytypes.HTTPNotificationDriver

# URIs of the form "tcp:..." interpreted as an TCP URL;
# e.g. "tcp://connecterra.com:5000"
# The notification is sent in XML to the specified URL using a client TCP socket.
com.connecterra.ale.notificationDriver.tcp.class =
com.connecterra.ale.notifytypes.TCPNotificationDriver

# JMS Driver's fully specified class name
#com.connecterra.ale.notificationDriver.jms.class =
com.connecterra.ale.notifytypes.JMSNotificationDriver

# The location of the naming.props file to be used by the driver.  The value of this
# property should be either a fully qualified file name or a file name relative to the
# location of this edge.props file, # e.g. ../etc/naming.props
#com.connecterra.ale.notificationDriver.jms.default.namingPropertiesFile =

# The javax.naming.Context implementation that this driver will use perform JNDI
# lookups. If unspecified or if the specified class cannot be loaded, then a default of
# javax.naming.InitialContext will be used
#com.connecterra.ale.notificationDriver.jms.default.namingInitialContextClass =

# URIs of the form "file:///..." interpreted as a file URL;
# e.g., “file:///user/joe/myfile"
#
# File URIs can be:
# - an existing file, to which the notification is appended
# - an existing directory, in which each notification is written as a separate file
#   with a filename of the form <specName>-<timestamp>.xml, where specName is the
#   ECSpec name and timestamp is yyyyMMddhhmmssSSS
# - a pattern (see documentation)
#
# In all cases, the notification is written as XML
com.connecterra.ale.notificationDriver.file.class =
com.connecterra.ale.notifytypes.FileNotificationDriver

# URIs of the form "null:..."
# This driver does nothing, it just silently absorbs reports.
com.connecterra.ale.notificationDriver.null.class =
com.connecterra.ale.notifytypes.NullNotificationDriver
```

```
# ================================================================
#
# Reader properties
#
# These properties describe the configuration of the physical readers and the queues
# which manage them.
#

# The maximum number of threads that will simultaneously attend to readers
com.connecterra.ale.readerConcurrency = 10

# If true, the engine will not start when there is an incorrect reader configuration.
# If false, the engine will ignore the particular reader with the incorrect
# configuration during initialization.
com.connecterra.ale.reader.stopOnIncorrectReaderConfig = true

# For each physical reader to be defined, include a set of properties like this:
#
# com.connecterra.ale.reader.<deviceID>.class = <className>
# com.connecterra.ale.reader.<deviceID>.<prop1> = <prop1value>
# com.connecterra.ale.reader.<deviceID>.<prop2> = <prop2value>
# ...
# where prop1, prop2, etc, are defined by the reader class.
#
# It is up to each physical reader class to decide what logical readers are defined.
# For the common case where a physical reader class defines a single logical reader, it
# should provide a property like this:
#
# com.connecterra.ale.reader.<deviceID>.logicalReaderName = <logicalReaderName>

 com.connecterra.ale.reader.SimReadr.class =
com.connecterra.ale.readertypes.ThingMagicMercury4PhysicalReader
 com.connecterra.ale.reader.SimReadr.hostname = localhost
 com.connecterra.ale.reader.SimReadr.port = 5050
 com.connecterra.ale.reader.SimReadr.defaultRate = 0
 com.connecterra.ale.reader.SimReadr.uhf2LogicalReaderName = ConnecTerra2
 com.connecterra.ale.reader.SimReadr.uhf1LogicalReaderName = ConnecTerra1

# A composite reader name allows a list of reader names (logical or composite reader
# names) to be associated with a single group reader name.  A composite reader name may
# be specified as a logical reader name in an ECSpec.
#
# For each composite reader name to be defined, include a set of properties like this:
#
# com.connecterra.ale.compositeReader.<composite reader name>.members = <reader name>[
<reader name>]*
# com.connecterra.ale.compositeReader.<composite reader name>.reportFailedMembers =
[true | false] (Default is false).
#
# Cyclic member specifications among composite reader definitions are not allowed and
# will be detected.  The reportFailedMembers property indicates whether or not
# ECReports will report decomposed member names as failed logical readers. If the
# property is set to false, ECReports will report composite reader names as failed
# logical readers.  The default is false.
#
# Member list for composite reader name CompositeReader1:
# com.connecterra.ale.compositeReader.CompositeReader1.members = ConnecTerra1
ConnecTerra2
```

```
# Each logical reader may also have an ordered list of filters associated with it.
# Each set of tags which is read by that logical reader is passed through the filter
# list, which may modify the tags which are actually returned to the application.
#
# For each filter to be defined, include a set of properties like this:
#
# com.connecterra.ale.filter.<filter name>.class = <className>
# com.connecterra.ale.filter.<filter name>.<prop1> = <prop1value>
# com.connecterra.ale.filter.<filter name>.<prop2> = <prop2value>
#
# Then, for each logical reader to which filters are applied, include a property
# like this:
#
# com.connecterra.ale.logicalReader.<logical reader name>.filters = <filter name>
[<filter name>]*
#
# For example:
#
# com.connecterra.ale.filter.transients.class =
com.connecterra.ale.filtertypes.TransientFilterFactory
# com.connecterra.ale.filter.transients.minReads = 3
# com.connecterra.ale.filter.transients.firmInterval = 1400
# com.connecterra.ale.filter.transients.expiredInterval = 1400
#
# com.connecterra.ale.logicalReader.MyReader.filters = transients
```

Setting Up Readers

You can configure RFTagAware to communicate with supported readers in one of two ways:

- Editing the `edge.props` file directly to configure these devices. This is the default for RFTagAware installations prior to version 1.3.
- Editing the reader configuration information using the Administration Console. This is the default for new RFTagAware 1.3 installations.

The default behavior is controlled by a property in the `edge.props` file called `com.connecterra.ale.dynamicConfig.enabled`. For new installations, this property is set to `true`, allowing the user to configure logical and composite readers from the Administration Console. (See RFID Devices on page 3-24 for more information.) If you wish to add and configure readers by editing the `edge.props` file, set this property to `false`.

> **Note:** If the `com.connecterra.ale.dynamicConfig.enabled` property is set to `true`, the persistence flag (`com.connecterra.ale.persistence.disabled`) should be set to `false` to allow reader configuration information to persist across invocations of the Edge Server.

Reader Simulator Configuration

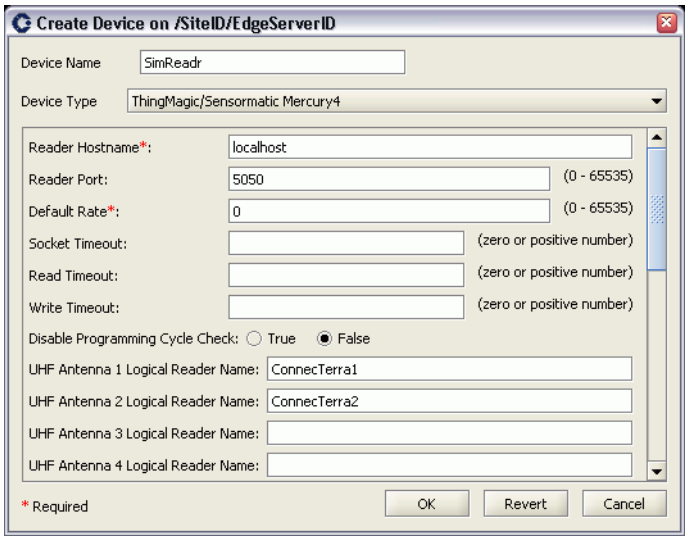
The `edge.props` file initially installed on your system contains a block of properties (shown below), defining a ThingMagic Mercury4 reader with two antennas running on your local system on port 5050. This set of properties refers to the RFTagAware Reader Simulator, which provides a minimal simulation of a ThingMagic Mercury4 reader or (with minor configuration) a Printronix printer.

```
com.connecterra.ale.reader.SimReadr.class =
com.connecterra.ale.readertypes.ThingMagicMercury4PhysicalReader
com.connecterra.ale.reader.SimReadr.hostname = localhost
com.connecterra.ale.reader.SimReadr.port = 5050
com.connecterra.ale.reader.SimReadr.defaultRate = 0
com.connecterra.ale.reader.SimReadr.uhf2LogicalReaderName = ConnecTerra2
com.connecterra.ale.reader.SimReadr.uhf1LogicalReaderName = ConnecTerra1
```

Note that a single `edge.props` file may contain properties definitions for many readers. For testing and development purposes, you may want to keep some definitions (for example, for the simulator) in the file but inactive. You can prevent RFTagAware from trying to communicate with a reader by simply commenting out the first line of a reader's property definitions. (To comment out a line, place a # at the beginning of the line.)

If you choose to configure the Reader Simulator through the RFID Devices pane on the Administration Console, you will need to enter properties into the Reader Configuration dialog as shown below. See RFID Devices on page 3-24 for instructions on configuring readers, and Configuring the Reader Simulator on page 4-3 for information on configuring the Reader Simulator.

Reader Configuration for the Reader Simulator.



Adding Additional Physical Readers

If you have chosen to configure readers through the RFID Devices pane on the Administration Console, see RFID Devices on page 3-24 for full instructions on configuring readers. You may also need to refer to the *RFTagAware Reader Configuration Guide* for details on each reader's properties.

The remainder of this section describes the reader configuration parameters used to configure additional readers in `edge.props`.

Within the `edge.props` file, each reader is configured by including several related lines of parameter definitions. Every physical reader includes a line having the following form:

```
com.connecterra.ale.reader.<physical reader name>.class = <device type name>
```

- `<physical reader name>` is whatever name you want to assign to the reader being configured. The name must be unique among all physical reader name assignments within the `edge.props` file, and must consist only of alphanumeric characters, hyphen (-), underscore (_), and plus sign (+).

  The physical reader name is used to refer to the reader within the RFTagAware Administration Console. It also appears in the `physicalReaderNames` list that is returned as a part of event cycle reports that include data from this reader.

- `<device type name>` specifies the name of a driver provided by ConnecTerra for the specific make and model of the reader. This name is in the form:
  `<reader_identifier>PhysicalReader`

Accompanying each instance of the above `edge.props` line is a block of additional properties that are specific to that particular reader. The properties required within each configuration block depend on the reader make and model – the particular properties required by each reader type are listed in the *RFTagAware Reader Configuration Guide*.

The properties for a given physical reader are identified by sharing a common `<physical reader name>` within their property names.

All physical reader configuration blocks contain definitions of the logical readers associated with a given physical reader. Each provisioned physical reader has one or more logical readers associated with it, one logical reader for each operational antenna. The definition of a logical reader specifies the logical reader's name, which is used when identifying the logical reader within an event cycle specification.

Each reader configuration line in `edge.props` looks like this:

```
com.connecterra.ale.reader.<phys reader name>.<property name>=<property value>
```

where `<phys reader name>` is the same for all properties corresponding to a particular physical reader, `<property name>` is the name given in the first column of a table below, and `<property value>` is the value you want to assign for that property. If a property is identified as optional, you can omit the corresponding line in the `edge.props` file and the default value will be used instead.

Using Composite Readers

You specify names for logical readers (i.e., read or write points) when defining physical readers. You can create additional logical readers by combining existing logical readers. A logical reader created in

this way is called a *composite reader*. By defining composite readers, you can decouple applications from decisions you take at deployment time about how many readers are needed to cover a single location.

For example, suppose that today, you have four logical readers covering a location called LoadingDock23:

- `LoadingDock23_Reader1`
- `LoadingDock23_Reader2`
- `LoadingDock23_Reader3`
- `LoadingDock23_Reader4`

You specify these reader names in each `ECSpec` that you create for LoadingDock23.

Then suppose you discover that you really need five readers to cover LoadingDock23. If you specified single logical reader names in each `ECSpec`, then you would need to:

- Add the fifth reader, using the `edge.props` file, or the Administration Console RFID Devices node.
- Go back and edit each `ECSpec` to include the new fifth reader.

Changing every `ECSpec` is undesirable, especially if some applications generate `ECSpec` instances for LoadingDock23 on the fly.

The alternative is to define a composite reader called `LoadingDock23`. Initially, this composite reader is configured to contain `LoadingDock23_Reader1` through `LoadingDock23_Reader4`. Applications that want to read from Loading Dock 23 simply specify `LoadingDock23` as the sole logical reader in their `ECSpec` instances.

Then, when you add your fifth reader, all you have to do is:

- Edit the `edge.props` file or use the Administration Console RFID Devices node to:
  - Add the fifth reader.
  - Change the definition of the `LoadingDock23` composite reader to include the fifth reader.
- Leave your `ECSpec` instances unchanged.

For information on how to define composite readers, see below.

Defining Composite Readers

If you have chosen to configure composite readers through the Composite Readers pane on the Administration Console, see Composite Readers on page 3-31 for full instructions on configuring readers. The remainder of this section describes how to define composite readers in `edge.props`.

Use the following lines in `edge.props` to define composite readers:

- `com.connecterra.ale.compositeReader.composite_reader_name.members = logical_reader_1 logical_reader_2`

  `composite_reader_name`: The name you want to use for this composite reader.

  `logical_reader_1 logical_reader_2`: List the logical (or composite) readers that make up this composite reader. Put a space between each reader name.

  Example:
  ```
  com.connecterra.ale.compositeReader.LoadingDock23.members=
  LoadingDock23_Reader1 LoadingDock23_Reader2 LoadingDock23_Reader3
  LoadingDock23_Reader4
  ```

  Note that cyclic member specifications among composite reader definitions are not allowed and will be detected.

- `com.connecterra.ale.compositeReader.composite_reader_name.reportFailedMembers = false`

  `composite_reader_name`: The name assigned in the statement above to identify this composite reader.

  When a reader fails, you can have `ECReports` include either the name of the composite reader that the failed reader belongs to OR the logical reader name of the failed reader.

  `true`: Lists failed readers by logical reader name.

  `false`: Lists failed reader by composite reader name. This is the default.

Using 64-bit Tags

RFTagAware supports the following 64-bit tag formats as defined by the *EPCglobal Tag Data Standards* Version 1.1:

- General Identifier (GID-I and GID-III)
- Serialized Global Trade Item Number (SGTIN)
- Serial Shipping Container Code (SSCC)
- Serialized Global Location Number (SGLN)
- Global Returnable Asset Identifier (GRAI)
- Global Individual Asset Identifier (GIAI)

When encoded onto 64-bit tags, these tag formats require an external translation table to translate the EPC Company Prefix Index field of the tag into an EAN.UCC Company Prefix. EPCGlobal supplies a translation table at this URL (that RFTagAware uses by default):

```
http://www.onsepc.com/ManagerTranslation.xml
```

Note: RFTagAware installations include a copy of this file in the `etc` subdirectory.

Alternatively, you can specify a local file as the source of the translation table by changing the value of `com.connecterra.ale.epcIndexTableURL` in `edge.props` to `file:///mydir/myfile.xml`, where `myfile.xml` has the general form:

```
<GEPC64Table>
<entry index="1" companyPrefix="0037000"/>
<entry index="2" companyPrefix="0041333"/>
...
<entry index="16383" companyPrefix="0614141"/>
</GEPC64Table>
```

If a file name is specified without an absolute path, the path given is assumed to be relative to RFTagAware's `etc` subdirectory. You may also specify a list of URLs, separated by spaces:

```
com.connecterra.ale.epcIndexTableURL=http://www.onsepc.com/
ManagerTranslation.xml file:///c:/mydir/myfile.xml
```

The Edge Server will try each URL in turn, from left to right, until it successfully reads from one of the URLs. You can use this syntax to specify the EPCglobal table as the primary source, and use the local file syntax as a backup in case the EPCglobal site is unreachable.

Using 96-bit Tags

RFTagAware supports the following 96-bit tag formats as defined by the *EPCglobal Tag Data Standards* Version 1.1. These tag formats do not require an external translation table.

- General Identifier (GID)
- Serialized Global Trade Item Number (SGTIN)
- Serial Shipping Container Code (SSCC)
- Serialized Global Location Number (SGLN)
- Global Returnable Asset Identifier (GRAI)
- Global Individual Asset Identifier (GIAI)

Specifying State Data Persistence

You can configure the Edge Server to either keep or discard state data each time you stop and start the Edge Server. "State data" means:

- The `ECSpec` instances, `PCSpec` instances, `EPCCacheSpec` instances, and subscribers that you create through the ALE API.
- `ECSpec`, `PCSpec`, `EPCCacheSpec` information including the number of subscribers for a given `ECSpec`, `PCSpec`, or `EPCCacheSpec`, the number of times it has been activated, and the date/time of last activation and report delivery.
- Reader configuration data for readers configured using the Administration Console.

You configure how you want to handle state data by editing `edge.props`, as described below.

The Edge Server also includes a utility that removes all state data. For information on how to use this utility, see Using ClearEdgeServerState to Delete State Data on page 2-27.

Editing edge.props to Configure State Data

Use the following lines from `edge.props` to tell the Edge Server how you want to handle state data:

`com.connecterra.ale.persistence.disabled = false`	This line tells the Edge Server whether or not to keep state data each time you stop and start the Edge Server. `true`: Discard state data. `false`: Keep state data.
`com.connecterra.ale.persistence.directory = ../var/edgestate`	Relative path to the directory where you want the Edge Server to store state data.
`com.connecterra.ale.persistence.allowBadData = false`	When it starts, the Edge Server may discover that the state data you have is incomplete or not properly formatted. When this happens, you can tell the Edge Server to either stop, or continue starting with the bad data. `true`: Continue Edge Server startup, using as much data as it can, and moving the bad data aside. `false`: Stop startup if the Edge Server encounters bad state data.
`com.connecterra.ale.persistence.interval = 0`	How frequently to save different types of state data. Can be zero or a non zero positive integer (milliseconds). Zero: Tells the Edge Server to save state data ONLY when: • An `ECSpec` is defined or undefined. • An `ECSpec` is subscribed or unsubscribed. • An `ECSpec` delivers reports. A non-zero positive integer: Tells the Edge Server to: • Save state data under the conditions listed for zero, AND • Save the activation count and last activated time periodically (according to the value of the integer), even when activations do not result in reports being generated.

Using ClearEdgeServerState to Delete State Data

The `ClearEdgeServerState` utility removes all state data that you are storing in your state data directory. After you run `ClearEdgeServerState`, your Edge Server will start with no `ECSpec` instances defined.

To run `ClearEdgeServerState`:

1. Stop the Edge Server.
2. Run `ClearEdgeServerState` as appropriate for your environment:

Windows: From a command line in the `control\bin` directory, type:
`ClearEdgeServerState.bat`

UNIX: From the RFTagAware installation directory, type:
`control/bin/ClearEdgeServerState.sh`

3. `ClearEdgeServerState` displays a confirmation message warning you that running `ClearEdgeServerState` will reset the Edge Server to the state it was in when it was newly installed, and that client applications that depend on Edge Server state may fail.
4. If you are sure you want to reset the Edge Server state, type:
`YES`

Note: If you run `ClearEdgeServerState` with the `-force` flag, then no confirmation prompt appears.

Configuring the HTTP Notification Driver

To configure the HTTP notification driver, you can use the following `edge.props` timeout values:

`connectTimeout`	Specifies, in milliseconds, how long the driver will wait to connect to an HTTP endpoint before failing. Defaults to 30000 milliseconds if not specified.
`readTimeout`	Specifies, in milliseconds, how long the driver will wait for a response before failing. Defaults to 30000 milliseconds if not specified.

Example:
```
com.connecterra.ale.notificationDriver.http.class =
com.connecterra.ale.notifytypes.HTTPNotificationDriver
com.connecterra.ale.notificationDriver.http.connectTimeout = 5000
com.connecterra.ale.notificationDriver.http.readTimeout = 10000
```

Enabling the SNMP Log Handler

RFTagAware installations as of version 1.3 support an SNMP Log Handler, which creates SNMP traps from RFTagAware log event messages and sends them to one or more configured destinations (called *trap sinks*). Typically, these destinations are SNMP-based Network Management Systems.

The default installation of RFTagAware does not enable the SNMP Log Handler. To enable it, use the following instructions to edit the `etc/logging.props.edge` file. This file contains Java property definitions that control which log messages are forwarded to which handlers.

1. Add the handler class (`com.connecterra.mgmtagent.SNMPLogHandler`) to the comma-separated list of handlers in `logging.props.edge`.
   `handlers = com.connecterra.mgmtagent.SNMPLogHandler`
2. Set the logging level associated with the SNMP Log Handler. We recommend setting the level to either WARNING or SEVERE to avoid excessive numbers of traps.
   `com.connecterra.mgmtagent.SNMPLogHandler.level= WARNING`
3. Configure one or more trap sinks as shown below. Each trap sink has a unique name assigned to it (e.g., `sink1` in the sample below), and up to four configuration properties. `Hostname` is a required property; the others (`port`, `snmpversion`, `community`) are optional.

```
# hostname is a required property when defining a trap sink.  It may be
# either a hostname or IP address.
com.connecterra.mgmtagent.SNMPLogHandler.trapSink.sink1.hostname =
myMgmtStation.acme.com

# port is an optional property.  The default SNMP UDP port is 162
com.connecterra.mgmtagent.SNMPLogHandler.trapSink.sink1.port = 162

# snmpversion specifies the version of the SNMP protocol SNMP traps
# directed to this trapSink will be encoded in.  This is an optional
# parameter, which if present must be set to either "v1" or "v2c".  The
# default value is "v2c".
com.connecterra.mgmtagent.SNMPLogHandler.trapSink.sink1.snmpversion = v2c

# community specifies the community string that appears in SNMP trap
# messages. This is an optional parameter; its default value is “public"
com.connecterra.mgmtagent.SNMPLogHandler.trapSink.sink1.community = public
```

4. Save your changes and restart the Edge Server. When log events of the type(s) specified are generated, SNMP trap information will be sent to the destination(s) you provided.

The ConnecTerra-specific MIB files can be found under `share/mibs` in the RFTagAware installation directory.

Setting up the JMS Notification Driver

To use the JMS notification driver, you need to:

- Edit various RFTagAware configuration files to include JMS specifications.

  See Editing Configuration Files for JMS on page 2-30.

- Set up a Java Naming and Directory Interface (JNDI) provider and JMS server.

  See Setting up the JNDI Provider and JMS Server on page 2-31.

Editing Configuration Files for JMS

To use the JMS notification driver, you need to edit the following Edge Server configuration files:

- `edge.props` (in the `etc` directory)
- `jms.options` (in the `etc` directory)
- `naming.props` (must be created in the `etc` directory when needed)

edge.props

Find the notification driver properties, and under `com.connecterra.ale.notificationDriver` (shortened to ND below), edit the JMS notification driver properties as shown.

Find and uncomment the `ND.jms.class` line.

```
com.connecterra.ale.notificationDriver.jms.class =
com.connecterra.ale.notifytypes.JMSNotificationDriver
```

Find and uncomment the `ND.jms.default.namingPropertiesFile` line, then set it to the file name of the JNDI naming properties file. The file name may include either an absolute path or a path relative to the location of `edge.props`.

```
com.connecterra.ale.notificationDriver.jms.default.namingPropertiesFile =
C:\\Program Files\\ConnecTerra\\RFTagAware\\1.3\\etc\\naming.props
```

See naming.props on page 2-31 for more information.

Find the `ND.jms.default.namingInitialContextClass` line. Setting this optional property configures the JMS Driver to use the specified class to perform JNDI lookups. The value of this class must be a valid Java class that is available in the system classpath. The value defaults to `javax.naming.InitialContext`, which is used when this property is not specified. Other values for this class are:

```
javax.naming.directory.InitialDirContext
javax.naming.ldap.InitialLdapContext
```

See the Javadoc for these classes when deciding which class to use.

jms.options

Edit the `JMS_LIB` environment variable to indicate the location of the JAR/ZIP files provided by your JMS vendor that provide the naming context factory class (as you previously specified in `edge.props`) and all other classes required by JMS clients for your JMS vendor's implementation. Vendor-specific samples of JMS options are in `/samples/JMSSamples`.

The exact convention for setting this environment variable depends on your operating system and shell environment.

Windows example: `set JMS_LIB=d:\programs\libraries\jmsclientlib.jar`
UNIX example: `JMS_LIB=/opt/lib/jmsclientlib.jar`

naming.props

The default name of the JNDI naming properties file is `naming.props`. You can specify a different name via the `com.connecterra.ale.notificationDriver.jms.default.namingPropertiesFile` setting in `edge.props`, if desired.

You will need to create this file in the `etc` directory. (It can be copied from any etc directory under `samples/JMSSamples/<vendor_name>/`.) The naming properties file is used to initialize the instance of the `javax.naming.Context` class. The name/value of the properties specified in the file are used as Context environment properties. For example:

```
java.naming.provider.url=iiop://host1:2809
java.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory
```

The properties in the `naming.props` file are considered default values, and can be overridden by a notification subscription URI (by adding the equivalent property to the notification URI as a query parameter). See the *RFTagAware Programmer Guide* for instructions on overriding these properties.

Vendor-specific examples of this file are in the RFTagAware installation directory under the `samples\JMSSamples\<vendor>\<sampleappname>\etc` directories.

Setting up the JNDI Provider and JMS Server

To use RFTagAware's JMS notification driver, you need to set up the following components in your environment:

- Java Naming and Directory Interface (JNDI)

  The RFTagAware JMS notification driver in the Edge Server uses the JNDI provider to obtain the necessary JMS objects for sending JMS messages. Examples of JNDI providers include:

  - LDAP provider
  - File system provider
  - JNDI provider built into an application server

  You need to configure the JNDI provider with a connection factory and queue name along with their corresponding JMS objects. Each vendor provides administrative tools for administering JMS objects.

- JMS server

  You need to configure the JMS server with the appropriate destinations (i.e., queue or topic). Each vendor provides administrative tools for configuring the JMS server.

This section includes configuration instructions for the following vendors:

- BEA® WebLogic Server Configuration (page 2-32)
- IBM® (page 2-32)

BEA® WebLogic Server Configuration

Use the BEA WebLogic Server Administration Console (for example, `http://<wlhost>:7001/console`) to configure the JNDI provider and JMS server.

For example, use the following Administration Console page flows:

- Configure JMS Server:

  Services, JMS, Servers, Configure a New JMS Server

  Create a JMS server. Example: `jms_server1`

- Configure JNDI Provider:

  Services, JMS, Servers, `jms_server1`, Configure Destinations, Configure a new JMS Queue

  Create a new queue with name (example: `TestQ`) and JNDI name (example: `jms/TestQ`)

  Note that BEA provides `weblogic.jms.ConnectionFactory` and `weblogic.jms.XAConnectionFactory` as default connection factories.

- Restart Application Server:

  Stop the application server and then start the application server to utilize the new configuration.

For additional information, see the WebLogic Server System Administration Documentation:
`http://e-docs.bea.com/wls/docs81/admin.html`

IBM®

WebSphere® Application Server with Embedded Messaging Configuration

Use the WebSphere Application Server Administration Console (example: `http://<was_host>:9090/admin`) to configure the Internal JMS Server and to configure a listener port.

For example, use the following Administration Console page flows:

- Configure Internal JMS Server:

  Servers, Application Servers, `server1`, Server Components, JMS Servers

  Add queue name (example: `TestQ`)

- Configure Listener Port:

  Servers, Application Servers, `server1`, Message Listener Service, Listener Ports, New

  Specify Listener Port Name (example: `JMSTestListener`)

  Specify Connection Factory JNDI name (example: `jms/TestQCF`)

  Specify Destination JNDI name (example: `jms/TestQ`)

- Configure WebSphere JMS Provider:

  Resources, WebSphere JMS Provider, WebSphere Queue ConnectionFactories, New, Create an object for ConnectionFactory "TestQCF" with JNDI name "jms/TestQCF".

  Resources, WebSphere JMS Provider, WebSphere Queue Destinations, New, Create an object for Queue "TestQ" with JNDI name "jms/TestQ".

- Apply Changes to Master Configuration:

  Click on Save link and then Save button to apply changes to master configuration.

- Restart Application Server:

  Stop the application server and then start the application server to utilize the saved master configuration.

WebSphere Application Server with Full WebSphere MQ Configuration

Use the WebSphere Application Server Administration Console (example: `http://<was_host>:9090/admin`) to configure a listener port.

For example, use the following Administration Console page flows:

- Configure Listener Port:

  Servers, Application Servers, `server1`, Message Listener Service, Listener Ports, New

  Specify Listener Port Name (example: `MQJMSTestListener`)

  Specify Connection Factory JNDI name (example: `jms/MQTestQCF`)

  Specify Destination JNDI name (example: `jms/MQTestQ`)

- Apply Changes to Master Configuration:

  Click on Save link and then Save button to apply changes to master configuration.

- Restart Application Server:

Stop the application server and then start the application server to utilize the saved master configuration.

WebSphere MQ Standalone Configuration

Use the WebSphere MQ Explorer to create a queue manager (example: `QM_host1`) and to create a queue (example: `MQ_JMS_Q`) under the queue manager.

Use the JMSAdmin tool to define connection factories and queues.

- Update `JMSAdmin.config` file with appropriate values for `INITIAL_CONTEXT_FACTORY` and `PROVIDER_URL` environment variables.
- Update JMSAdmin script to set and use the classpath with the JAR files in `Java/bin` of the WebSphere MQ installation directory.

For example, use the JMSAdmin tool to define a connection factory with JNDI name of `jms/MQTestQCF` and a queue with JNDI name of `jms/MQTestQ` as follows:

```
InitCtx> define qcf(jms/MQTestQCF) qmgr(QM_host1)
InitCtx> define q(jms/MQTestQ) qmgr(QM_host1) queue(MQ_JMS_Q)
```

Additional IBM Product Information

For additional information, see the IBM Product Publications:
`http://publib.boulder.ibm.com/index.html`

The following publications are recommended:

- IBM WebSphere Application Server and WebSphere MQ Family Integration Redbook (SG24-6878)
  `http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246878.html?Open`
- IBM WebSphere Application Server, Version 5.1: Getting Started (SC31-6323)
  `http://publibfp.boulder.ibm.com/epubs/pdf/c3163231.pdf`
- IBM WebSphere MQ: Using Java (SC34-6066)
  `http://publibfp.boulder.ibm.com/epubs/pdf/csqzaw12.pdf`
- IBM Tivoli Directory Server: Administration Guide (SC32-1339), located in the installation directory of the IBM Tivoli Directory Server
  `doc/enUS1252/admin_gd.pdf`

JBoss® Application Server Configuration

JBoss provides a default connection factory (with JNDI name of `ConnectionFactory`) which can be leveraged without any additional JNDI configuration.

For EJBs (i.e., Message Driven Beans configured to receive JMS messages) deployed in the JBoss application server, the JBoss-specific EJB deployment descriptor (`jboss.xml`) contains the

destination JNDI name (example: `TestQ`) to be configured. Note that JBoss automatically configures the destination JNDI name in the "queue" JNDI context for queues (example: `queue/TestQ`) or "topic" JNDI context for topics (example: `topic/TestTopic`).

For more advanced JMS or JNDI configurations, use the JBoss JMX administrative console (example: `http://<jboss_host>:8080/jmx-console`) to configure the JMS server and JNDI provider.

For more information, see the JBoss Administration and Development Documentation:
`http://jboss.org/docs/index`

Sun® Java System Application Server Configuration

Use the Application Server Admin Console (example: `http://<sun_host>:4848/asadmin`) to configure the JNDI provider and JMS server.

For example, use the following Administration Console page flows:

- Configure JMS Server and JNDI Settings:

  Java Message Service, JMS Hosts

  Java Message Service, Connection Factories, New

  Create a new connection factory with JNDI name (example: `jms/TestQCF`), type of `javax.jms.QueueConnectionFactory`, and resource enabled.

  Java Message Service, Physical Destinations, New

  Create a new physical destination with name (example: `PhysicalQueue`) and type of `queue`.

  Java Message Service, Destination Resources, New

  Create a new connection factory with JNDI name (example: `jms/TestQ`), type of `javax.jms.Queue`, and resource enabled.

  Add additional property called `Name` with value of `PhysicalQueue`.

- Restart Application Server:

  Stop the application server and then start the application server to utilize the new configuration.

For additional information, see the Sun Java System Application Server Administration Guide:
`http://docs.sun.com/db/doc/817-6088`

TIBCO® Enterprise for JMS Configuration

Use the JMS Administration Tool to create JMS connection factories and queues.

For example, after starting the JMS Server, use the JMS Administration Tool to define a connection factory with JNDI name of `TestQCF` and a queue with JNDI name of `TestQ` as follows:

```
> connect host1:7222
tcp://host1:7222> create factory TestQCF queue
tcp://host1:7222> create queue TestQ
```

For additional information, see the TIBCO Enterprise for JMS User's Guide, located in the installation directory of TIBCO Enterprise for JMS:

```
JMS/doc/pdf/tib_jms_users_guide.pdf
```

Starting and Stopping RFTagAware

You start RFTagAware with individual scripts in the directory:

```
RFTagAware_install_dir/control/bin
```

where `RFTagAware_install_dir` is the directory where you installed the RFTagAware software.

Note that in a custom installation, not all files may be on the same machine.

Starting RFTagAware in a Windows Environment

From a command line in the `control\bin` directory, type these commands in the order shown below:

- `RunAdminConsole.bat`

  (Brings up the Administration Console GUI, as described in Chapter 3: Using the Administration Console.)

- `RunReaderSim.bat`

  (Optional – type only if you are using the reader simulator software, rather than actual readers.)

- `RunEdgeServer.bat`

  (Starts the Edge Server. Displays a console window on the machine where you started the Edge Server.)

  Because it is installed as a Windows service, you can also start the Edge Server through the Control Panel:

  Start, Settings, Control Panel, Administrative Tools, Services, right click Edge Server, then select Start.

- `RunQuickTest.bat`

  (Optional – Provides a quick test to see that your Edge Server and readers are operating correctly. See Running the QuickTest Utility on page 2-38 for details. Note that in either a standard or custom installation, the QuickTest utility is automatically installed in the same location as the Edge Server.)

Edge Server as a Windows Service

In a Windows environment, the Edge Server is installed as a Windows service. By default it is installed in manual, stopped mode. To alter the default, open the Component Services control panel, found under the Administrative Tools option on the Control Panel window. Choose the Services (Local) window in the left pane, and double-click the RFTagAware Edge Server service to access the service's properties.

You can start the Edge Server either by starting the Windows service, or by running the batch file, `RunEdgeServer.bat` (see Starting and Stopping RFTagAware on page 2-36). The batch file runs the Edge Server as a standalone process. It does not start the service in place. If you try to run it standalone while the service is running, or vice versa, it will fail.

When running the Edge Server as a Windows service, the `edge.wrapper.conf` file must be updated with `wrapper.java.classpath` properties set to the JMS implementation jar files (specified in `jms.options`).

Starting RFTagAware in a UNIX Environment

From the RFTagAware installation directory, type these commands in the order shown below:

- `control/bin/RunAdminConsole.sh`

  (Brings up the Administration Console GUI, as described in Chapter 3: Using the Administration Console.)

- `control/bin/RunReaderSim.sh`

  (Optional – type only if you are using the reader simulator software, rather than actual readers.)

- `control/bin/RunEdgeServer.sh`

  (Starts the Edge Server. Displays a console window on the machine where you started the Edge Server.)

- `control/bin/RunQuickTest.sh`

  (Optional – Provides a quick test to see that your Edge Server and readers are operating correctly. See Running the QuickTest Utility on page 2-38 for details. Note that in either a standard or custom installation, the QuickTest utility is automatically installed in the same location as the Edge Server.)

Starting the Edge Server at System Boot

In a Unix environment, the RFTagAware installation includes an initialization script for the Edge Server. This script (called `rftagaware`) is installed into the `bin` directory. It can be used to ensure that the system starts the Edge Server at system boot, and restarts the Edge Server automatically in the case of an unexpected exit.

The script takes one argument: `start`, `stop`, or `restart`.

This Argument...	Fails If...
`start`	If the Edge Server fails to start three times in a row. If the Edge Server is already running.
`stop`	If the Edge Server process cannot be killed. (Warn only) If the Edge Server is not running.
`restart`	If stop or start fails.

The `rftagaware` script can be run as needed from within the RFTagAware install directory, or invoked on system startup. If it will be a startup script, you will need to copy it to the `/etc/init.d` directory and create a symbolic link to it in one of the system runlevel directories (`/etc/rc[0-6].d`) before it will run automatically.

Running the QuickTest Utility

Once you have set up your readers and started the Edge Server, you can run the QuickTest utility to check that your Edge Server is operating correctly. The QuickTest utility establishes contact with the Edge Server, and then attempts to read from each of the logical readers defined (via `edge.props` or dynamically configured using the Administration Console).

- If it succeeds in reading from a reader, the utility identifies each EPC tag that the reader sees.
- If it fails to communicate either with a reader or with the Edge Server, the QuickTest utility briefly describes the problem, writes debugging information to a log file, and provides you with information to help diagnose the problem.

Starting the QuickTest Utility

The QuickTest utility is automatically installed in the same location as the Edge Server. Start the utility with one of the following:

- Windows

  From a command line in the `control\bin` directory, type:
  `RunQuickTest.bat`

- UNIX

  From the RFTagAware installation directory, type:
  `control/bin/RunQuickTest.sh`

Sample QuickTest Output

Below is sample output from a successful run of the QuickTest utility. In this example, the Edge Server is connected to two readers. The first reader sees no tags. The second reader sees six EPC tags.

```
C:\Program Files\ConnecTerra\RFTagAware\control\bin>RunQuickTest.bat
ConnecTerra(tm) RFTagAware (tm)
Copyright (C) 2004, 2003 ConnecTerra, Inc.
All Rights Reserved; use is subject to license terms.
US and International Patents Pending

Connecting to edge server...
Finished connecting to edge server.

Testing logical reader Connecterra1...
   Logical Reader Connecterra1 read no EPCs.
Finished testing logical reader ConnecTerra1.

Testing logical reader ConnecTerra2...
   Logical reader Connecterra2 read the following 6 EPCs:
   urn:epc:tag:gid-64-i:38000.4971.7
   urn:epc:tag:gid-64-i:38000.871.3
   urn:epc:tag:gid-64-i:38000.1571.4
   urn:epc:tag:gid-64-i:38000.2971.6
   urn:epc:tag:gid-64-i:38000.571.2
   urn:epc:tag:gid-64-i:38000.171.1
Finished testing logical reader ConnecTerra2.

Press any key to continue
...
```

Stopping RFTagAware

To shut down the software, either terminate or kill each of the three processes you started in Starting RFTagAware in a Windows Environment on page 2-36 or Starting RFTagAware in a UNIX Environment on page 2-37.

Maintaining the RFTagAware System

Backing Up the System

System backups should be planned in the initial architectural design phases of your site's RFTagAware-based implementation. Frequent backups and offsite storage are two key components of a successful system administration strategy.

At a minimum, backups should include the following directories under *RFTagAware_install_dir/RFTagAware_version/*, where *RFTagAware_install_dir* is the directory where you installed the RFTagAware software, and *RFTagAware_version* is the directory where the latest version of RFTagAware is installed on your system:

- `etc/` – This directory contains configuration information that can be used to configure a new Edge Server in the event of a machine failure.
- `var/` – This directory contains state and log information stored by the Edge Server.

Caution: If your backup plan requires backing up a running Edge Server, make certain beforehand that your solution supports backing up open files.

- `edgestate/` – This persistence store maintains the Edge Server state data about `ECSpec`, `PCSpec`, and `EPCCacheSpec` instances and their subscribers, and reader configuration data for readers configured using the Administration Console. The type and number of files in this directory, and the frequency of file updates, depend on how your Edge Server is deployed, and for what purposes. The location of this directory is configurable.

  The persistence store is configured in `edge.props`. The settings in this file specify the location of the persistence store, whether or not you are keeping state data each time you stop and restart the Edge Server, and other configuration details. For more information, see Editing edge.props to Configure State Data on page 2-27.

While not required, you may also consider adding the `bin` or `lib` directories as backup targets, if either directory contains files that can not be fully recreated by the RFTagAware installer.

Restoring the System

The procedures for restoring your Edge Server vary based on the condition of the hardware and availability of the persistence store. The general steps for restoring a failed system are:

1. Obtain and reinstall the proper version of RFTagAware.
2. Restore data from the backup server to the new RFTagAware host.
3. If the persistence store (typically `var/edgestate/`) on the failed system is not available, applications will need to define new `ECSpec` or `PCSpec` and `EPCCacheSpec` instances and subscribe to them.

In the following instructions, assume the failed Edge Server hardware is E1 and the replacement Edge Server hardware is E2:

1. Assign E2 the same hostname, and IP address, as E1.
2. Install RFTagAware on E2. This must be the same version that previously installed on E1.
3. Copy the backed-up configuration (`etc/*`) data and persistence store (`var/edgestate/*`) data (if available) to the same location on E2 relative to the RFTagAware install directory.
4. Start E2.
5. If persistence store data was not available:
   - For applications that read tags, have each application define new `ECSpec` instance(s).
   - For applications that write tags, have each application define new `PCSpec` instance(s) and `EPCCacheSpec` instance(s).
   - Application(s) subscribe to `ECSpec`, `PCSpec` or `EPCCacheSpec` instances on E2.
   - R-configure readers previously configured via the Administration Console on E2.

What's Next: Developing for the ALE API

You can now start your own development, using the ALE API. For complete reference information, sample applications, and code walkthroughs, see the *RFTagAware Programmer Guide*.

Chapter 3: Using the Administration Console

Contents

This chapter describes how to use the RFTagAware Administration Console.

Overview

The RFTagAware Administration Console is a graphical Java application that lets you:

- Monitor the activities of multiple Edge Servers and readers.

  See Monitoring Edge Servers on page 3-4 and Monitoring Readers on page 3-28.

- Add, remove, and configure readers and composite readers

  See RFID Devices on page 3-24 and Composite Readers on page 3-31

  Note that you can only configure readers if you are running an Edge Server installed with RFTagAware 1.3 or higher.

- View and edit a list of `ECSpec` data objects and their subscribers.

  See ECSpecs on page 3-8 and ECSpec Editor on page 3-14.

  Note that you can only edit ECSpecs if you are running an Edge Server installed with RFTagAware 1.2 or higher.

Starting and Stopping the Administration Console

Prerequisite: RFTagAware software must be installed on your system, as described in Chapter 2: Installing and Configuring RFTagAware.

How to Start

See Starting and Stopping RFTagAware on page 2-36.

How to Stop

From within the Administration Console, click **File, Exit**.

Administration Console Menus

The Administration Console has three main menus:

- File Menu (page 3-3)
- View Menu (page 3-4)
- Help Menu (page 3-4)

File Menu

File Menu	

Configure EdgeServers	Lets you add an Edge Server to the Administration Console. For each Edge Server, you need to specify:
	Edge Server URL: The location where the Edge Server is running. Use the format:
	http://*host_name*:*port*
	where *host_name* is the host name of the machine where the Edge Server is running, and *port* is the port the Edge Server is listening on (configured when you installed the Edge Server).
	If you cannot remember the port you configured, you can find it in `etc/edge.props`:
	`# The TCP port on which the ALE service listens.`
	`com.connecterra.ale.servicePort = 6060`

Preferences	**AutoRefresh Interval** How often to contact the Edge Server to update the ECSpec information (shown in the ECSpecs Pane on page 3-8) displayed on the Administration Console (in milliseconds). Default: 5000 (5 seconds). **Telemetry Port** The port on which the Administration Console receives telemetry information from Edge Servers. Default: 9999 **Alert History Length** How many alerts to store for each device. Default: 2000
Exit	Quits the application.

View Menu

View Menu	
Refresh ECSpecs	This menu choice is active only if you have the ECSpecs node selected in the tree view for a particular Edge Server. The Administration Console will contact the Edge Server and will retrieve the list of all active `ECSpec` instances and additional information for each. This operation may take some time if there are many `ECSpec` instances with data to be retrieved. Normally the ECSpec information is automatically refreshed according to the value you set for the Auto Refresh Interval in the Preferences dialog. This option is useful if the auto-refresh interval is particularly long, or if you have made a change you want to see immediately.
Refresh Device Browser	Rebuilds the tree shown in Device Browser pane. The Administration Console will attempt to contact each of the Edge Servers defined. If a particular Edge Server cannot be contacted, then the device browser will show the host name and port number (for example, `localhost:6060`) and an error indicator. Note that this operation may take some time if there are many Edge Servers to scan, and if some of the Edge Servers are not responding.

Help Menu

Help	
Help	Displays online help for the Administration Console.
About	Displays version and copyright information.

Monitoring Edge Servers

You can monitor how many event cycles have completed since you started an Edge Server.

You can also examine configuration information for an Edge Server, and examine the messages and warnings generated by an Edge Server.

For details, see:

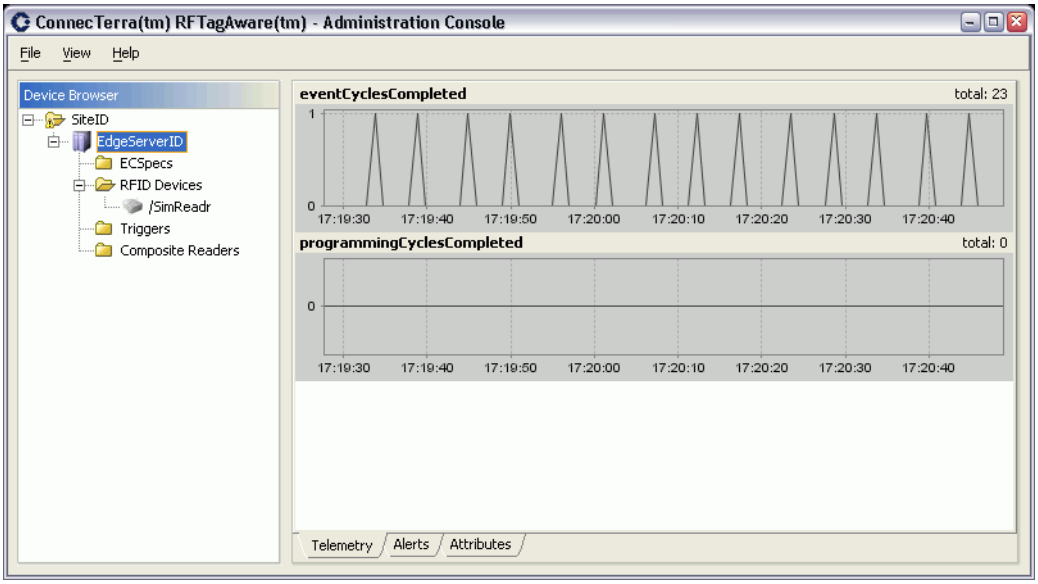
- Edge Server Telemetry (page 3-5)
- Edge Server Alerts (page 3-6)
- Edge Server Attributes (page 3-7)

Edge Server Telemetry

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, click the name of an Edge Server.
3. In the right panel, click the **Telemetry** tab (if it is not already selected).

Edge Server Telemetry Tab



The Edge Server Telemetry tab displays information about the number of event and programming cycles that have completed since you started the Edge Server.

eventCyclesCompleted

Shows when event cycles occurred.

total	The total number of event cycles completed since the Edge Server was started.
horizontal axis	Time stamps, in 24 hour time.
vertical axis	Shows how many event cycles were completed within the telemetry interval configured in the Edge Server. When the telemetry interval is very short, this value is likely to be either zero (no event cycles) or one (one event cycle). If the telemetry interval is longer, larger values will be displayed when many event cycles are completed within one telemetry interval.

programmingCyclesCompleted

Shows when programming cycles occurred.

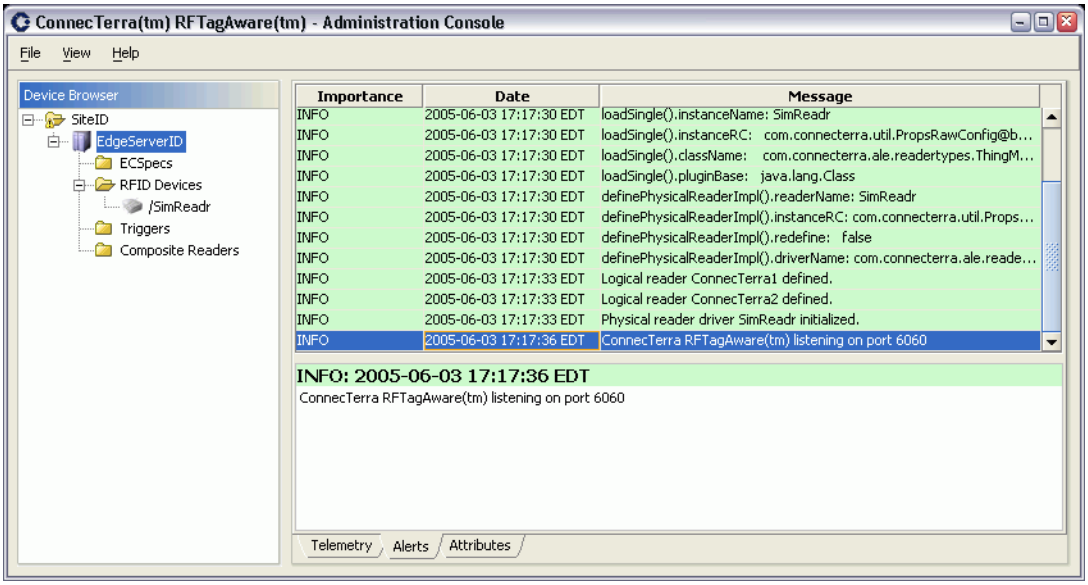
total	The total number of programming cycles completed since the Edge Server was started.
horizontal axis	Time stamps, in 24 hour time.
vertical axis	Shows how many programming cycles were completed within the telemetry interval configured in the Edge Server.

Edge Server Alerts

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left pane, click the name of an Edge Server.
3. In the right pane, click the **Alerts** tab.

Edge Server Alerts Tab



The Edge Server Alerts tab shows messages and alerts from the Edge Server.

When you click an alert in the top panel, a preview panel opens at the bottom of the tab, displaying the complete text of the alert.

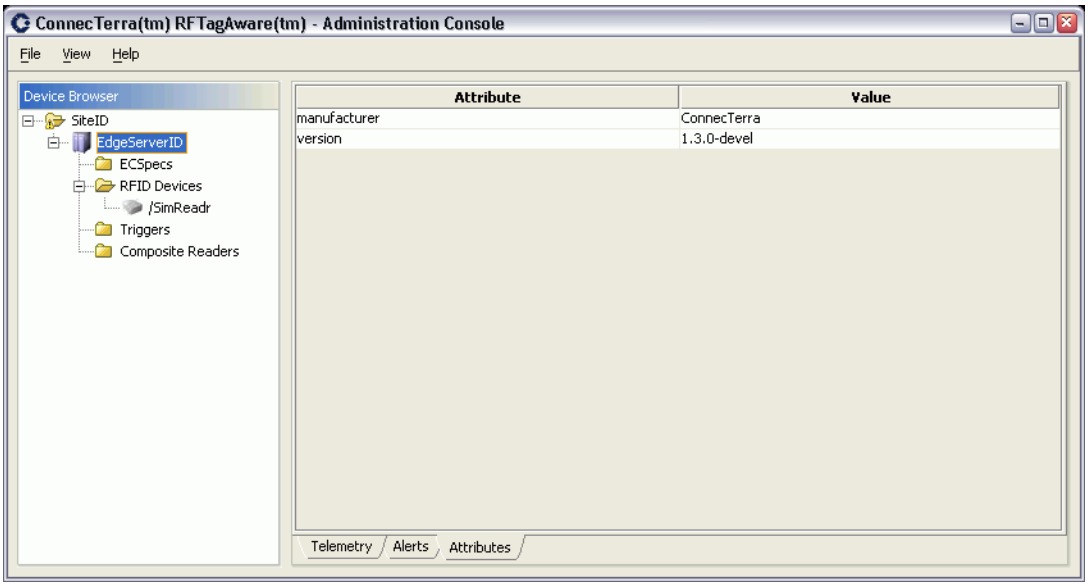
Importance	Can be: INFO (green) WARNING (yellow) SEVERE (red)
Date	The date and time the log entry was written.
Message	The text of the message.

Edge Server Attributes

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left pane, click the name of an Edge Server.
3. In the right pane, click the **Attributes** tab.

Edge Server Attributes Tab



The Edge Server Attributes tab shows configuration information for this Edge Server.

manufacturer	ConnecTerra
version	The version of the Edge Server.

ECSpecs

You use `ECSpec` data objects to tell the Edge Server what information you want from your readers, and how you want that reader information reported. `ECSpec` objects can get quite complex. The Administration Console's graphical user interface provides an easy way to quickly create and activate `ECSpec` objects, view the resulting reports, and modify your `ECSpec` design to meet the needs of the application you are developing.

For example, using the Administration Console GUI, you can easily:

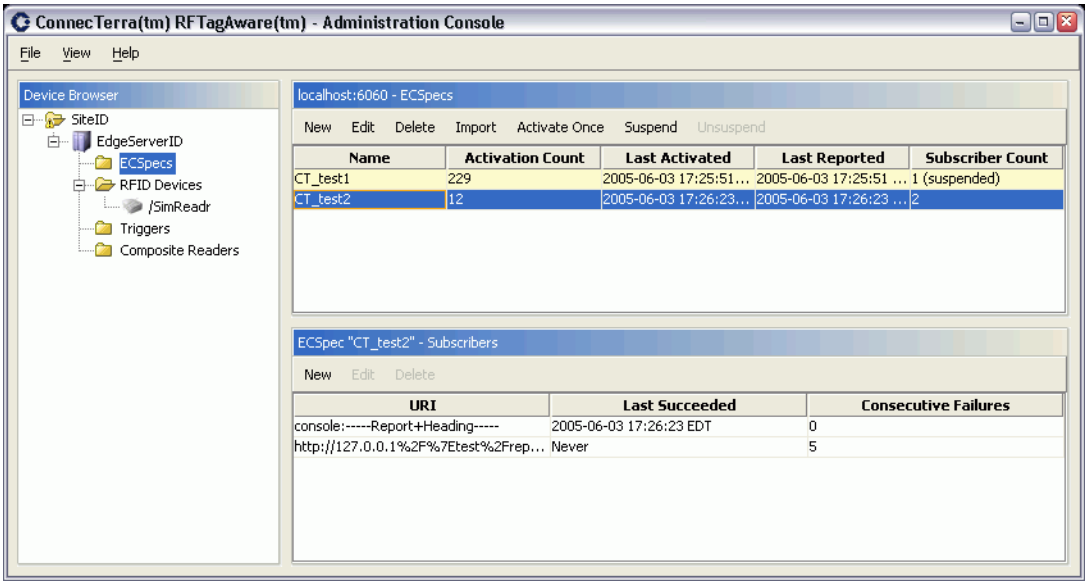
- Specify what data you want in a given report.
- Specify where to deliver the report, for example, an HTTP post to a web address, to a file on your system, or a display on the Edge Server console.
- Activate an ECSpec once and view the results.
- Examine the report to see if you are getting the information you want.
- Repeat these steps until you are satisfied with the results. When you are satisfied with the results you are seeing in the Administration Console, it is easy to export your `ECSpec` information for programmatic integration into a production system.

The details of `ECSpec` objects are covered in the *RFTagAware Programmer Guide*.

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left pane, expand the tree for an Edge Server.

ECSpecs Pane



3. Click the ECSpecs node under that Edge Server. The ECSpecs currently defined for this Edge Server are listed in the upper ECSpecs pane on the right.

The body of the ECSpecs pane is a list of ECSpecs and associate status information, described in the table below.

Name	The name of an `ECSpec`. You assign this name when you create the `ECSpec`.
Activation Count	An `ECSpec` is "activated" each time the start condition is met. This column shows how many times this `ECSpec` has been activated since its creation.
Last Activated	The last time this `ECSpec` was activated.
Last Reported	The last time this `ECSpec` generated a report. Not every ECSpec activation will produce a report.
Subscriber Count	The number of subscribers associated with this ECSpec. If the ECSpec has been suspended, a (Suspended) notation appears next to the number of subscribers.

The Name column is taken from the `ECSpec` object, while the other columns are taken from the `ECSpecInfo` object. For more information on these objects, see Chapter 5 of the *Programmer Guide*.

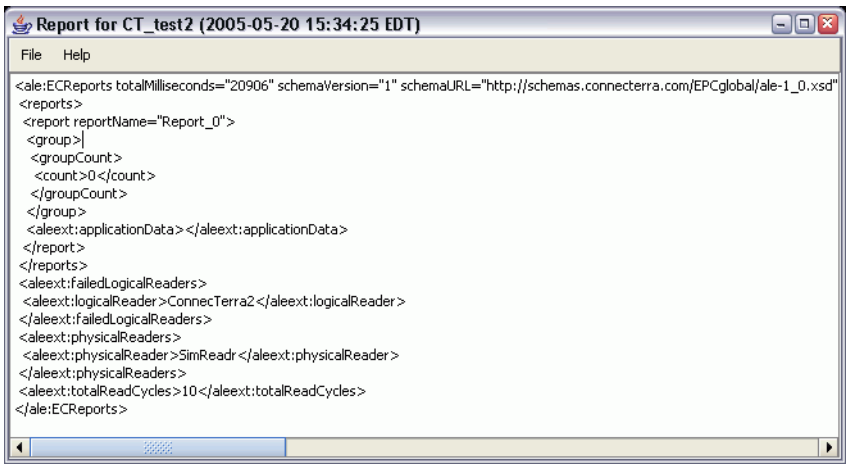
Using this pane, you can:

- add or edit ECSpecs (see ECSpec Editor on page 3-14 for more information)
- work with an ECSpec, including suspending/unsuspending, testing (via "Activate Once"), deleting, and showing subscribers
- add, remove, and edit subscribers associated with ECSpecs

Working with ECSpecs

To generate a test report for an ECSpec, click the ECSpec in the ECSpecs pane, then click the Activate Once button. A Report window will display the XML report data from the Edge Server.

Report Window



A report will appear after the ECSpec has completed one activation (i.e., the Start and Stop conditions have been met). If the report does not appear quickly, a progress dialog will appear. You can cancel the report from this dialog, if desired.

Note: Activating an ECSpec will turn on any readers used by that ECSpec.

The Report window contains menu options to export the report to a file (File | Export...), close the window (File | Close), and access online help (Help | EC Report Help).

To suspend an ECSpec, click the ECSpec in the ECSpecs pane, then click the Suspend button. While the ECSpec is suspended, no information is sent to any of the subscribers associated with that ECSpec. The Subscriber Count column on the ECSpecs pane displays a parenthetical notation if the ECSpec has been suspended.

Note: Suspending an ECSpec may turn off any readers used only by that ECSpec.

To unsuspend an ECSpec, click the ECSpec in the ECSpecs pane, then click the Unsuspend button. The ECSpec resumes sending information to subscribers, and the notation is removed from the Subscriber Count column on the ECSpecs pane.

Note: Unsuspending an ECSpec will turn on any readers used by that ECSpec when the Start condition is met.

To remove an ECSpec, simply click the ECSpec from the list on the ECSpecs pane, then click the Delete button. The ECSpec, all its reports, and all subscribers associated with that ECSpec are removed.

To show subscribers for a given ECSpec, click a row in the ECSpecs pane on the right. A list of subscribers (destinations for ECSpec report information) and associated status information appear in the bottom Subscribers pane, described in the table below.

URI	The URI for this subscriber.
Last Succeeded	The last time the Edge Server succeeded in sending a report to this subscriber.
Consecutive Failures	The number of consecutive times the Edge Server failed in its attempts to send a report to this subscriber.

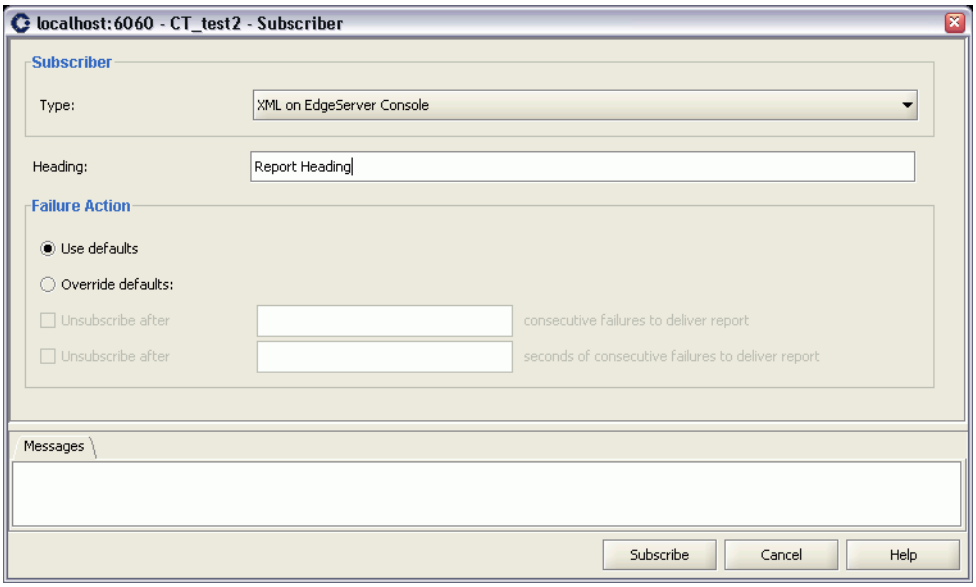
The URI column is taken from the `ECSubscription` object, while the other columns are taken from the `ECSubscriptionInfo` object. For more information on these objects, see Chapter 5 of the *Programmer Guide*.

Creating and Removing Subscribers

To create a new subscriber and associate it with the selected ECSpec:

1. Click an ECSpec on the ECSpecs pane.
2. Click the New button on the Subscribers pane. The Subscriber window displays.

Subscriber Window (New Subscriber)



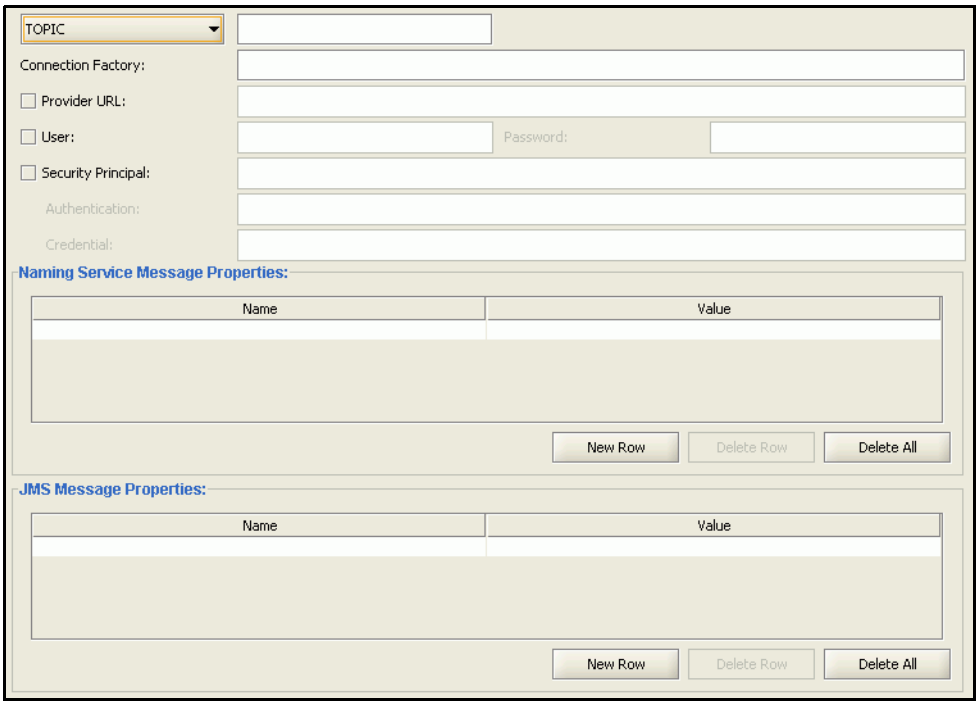
3. Choose a subscriber type from the list. The subscriber type determines what other fields appear in the window.
4. Fill in a destination for the report information to be sent to this subscriber. The table below describes the report destinations for the various subscriber types:

Subscriber Type	Destination Prompt(s)	Description
XML on Edge Server Console	"Heading"	Enter a text comment to write the report to the console in XML, preceded by the comment entered in the Heading field.
XML via HTTP POST	"http://"	Enter a URL to deliver the report using HTTP POST. Formats: `host:port/remainder-of-URL` (The colon and port number may be omitted; the port number defaults to 80.)
XML via JMS Message	"TOPIC" or "QUEUE"	Enter the name of the topic that the JMS notification driver will publish to, or the name of the queue that the JMS notification driver will add to. The topic or queue entered must exist in the JMS server.

Subscriber Type	Destination Prompt(s)	Description
XML File on Edge Server	"Directory or File"	Enter a directory path or file name to deliver the report by writing to the specified file.
TCP/IP	"Host" and "Port"	Enter a host name and port number to deliver the report using a client TCP socket.
Other	"URI"	Enter a URI to deliver the report to.

5. *(JMS only)* Fill in values for the subscriber-specific properties shown

Subscriber Window (JMS Options)



Name	Description
Connection Factory	The JNDI name of the connection factory (from the JMS server) for obtaining a topic connection or a queue connection. Use a topic connection factory name when publishing to a topic. Use a queue connection factory name when adding to a queue.
Provider URL	Optional. The URL for the JNDI Naming service.
User	Optional. User name used to create a JMS topic or queue connection.
Password	Optional. Password used to create a JMS topic or queue connection.
Security Principal	Optional. The security principal associated with the JNDI Naming Service.
Authentication	Optional. The authentication string associated with the security principal.
Credential	Optional. The security credential associated with the security principal.

Name	Description
Naming Service Message Properties	Optional. These are added to the `javax.naming.Context` environment when one is constructed to access a naming service to perform the necessary JNDI lookups. **Name** is the naming service message property name. Names are prefixed with `jndi:`. **Value** is the string value.
JMS Message Properties	Optional. These are added to the `javax.jms.TextMessage` as `String` properties. **Name** is the text message property name. **Value** is the string value.

6. Fill in values for the Failure Action section. If the delivery of a report to a subscriber fails on consecutive tries, the Edge Server can automatically unsubscribe that subscriber. The Failure Action section defines the conditions under which that can happen. You can choose to use the default failure action (Use defaults) or override it (Override defaults), by clicking on the appropriate radio button.

   If you chose to override the default, specify one or both failure actions by clicking on the check box next to the action – unsubscribe after N consecutive failures, or unsubscribe after N seconds of consecutive failures. You will also need to enter a reasonable number for each action selected.

7. Click the Subscribe button to save your changes and associate the new subscription with the ECSpec.

To remove a subscriber from an ECSpec:

1. Click an ECSpec on the ECSpecs pane.
2. Click a subscriber in the Subscribers pane.
3. Click the Delete button just above the list of subscribers.

Editing Subscribers

To edit a subscriber to the selected ECSpec:

1. Click an ECSpec on the ECSpecs pane.
2. Click a subscriber on the Subscribers pane.
3. Click the Edit button. The Subscriber window displays.
4. Make any desired changes and click the Save button to save them.

   Note: When the subscriber is saved, the original subscriber is unsubscribed, then the changed subscriber is saved. If the save fails, the old subscriber remains unsubscribed.

ECSpec Editor

Using the ECSpec Editor window, you can:

- add, import, edit, and deploy ECSpecs
- add and edit ECSpec reports, report filters, and report groups
- test an ECSpec
- export an ECSpec to an XML file

ECSpec Editor Menus

The ECSpec Editor has four main menus:

- File Menu (page 3-14)
- Tools Menu (page 3-14)
- View Menu (page 3-15)
- Help Menu (page 3-15)

File Menu

File Menu	
Deploy	This menu item is enabled if you have unsaved changes to a previously defined ECSpec. It deploys this ECSpec using the current name (equivalent to `redefine` in the ALE API).
Deploy As...	Deploy this ECSpec using another name. You are prompted for the name. This item is equivalent to `define` in the ALE API.
Revert	Reset the ECSpec being edited to the to the last deployed version, and discard any recent changes made using the ECSpec editor. If you have imported an ECSpec and have not yet deployed it, reset the ECSpec being edited to the version you imported.
Export	Export this ECSpec to an XML file.
Close	Close the Administration Console window.

Tools Menu

Tools Menu	
Test	Test this ECSpec. See Testing an ECSpec on page 3-17 for more information.

View Menu

View Menu	
ECSpec Editor	View the ECSpec in the ECSpec Editor. You can use the GUI controls here to modify and deploy your ECSpecs.
As XML	View the ECSpec in XML format. Use the ECSpec Editor view to make further changes.
As Text	View the ECSpec in text format. Use the ECSpec Editor view to make further changes.

Help Menu

Help	
ECSpec Help...	Displays online help for the ECSpec Editor.

Creating an ECSpec

To add a new ECSpec to the selected Edge Server:

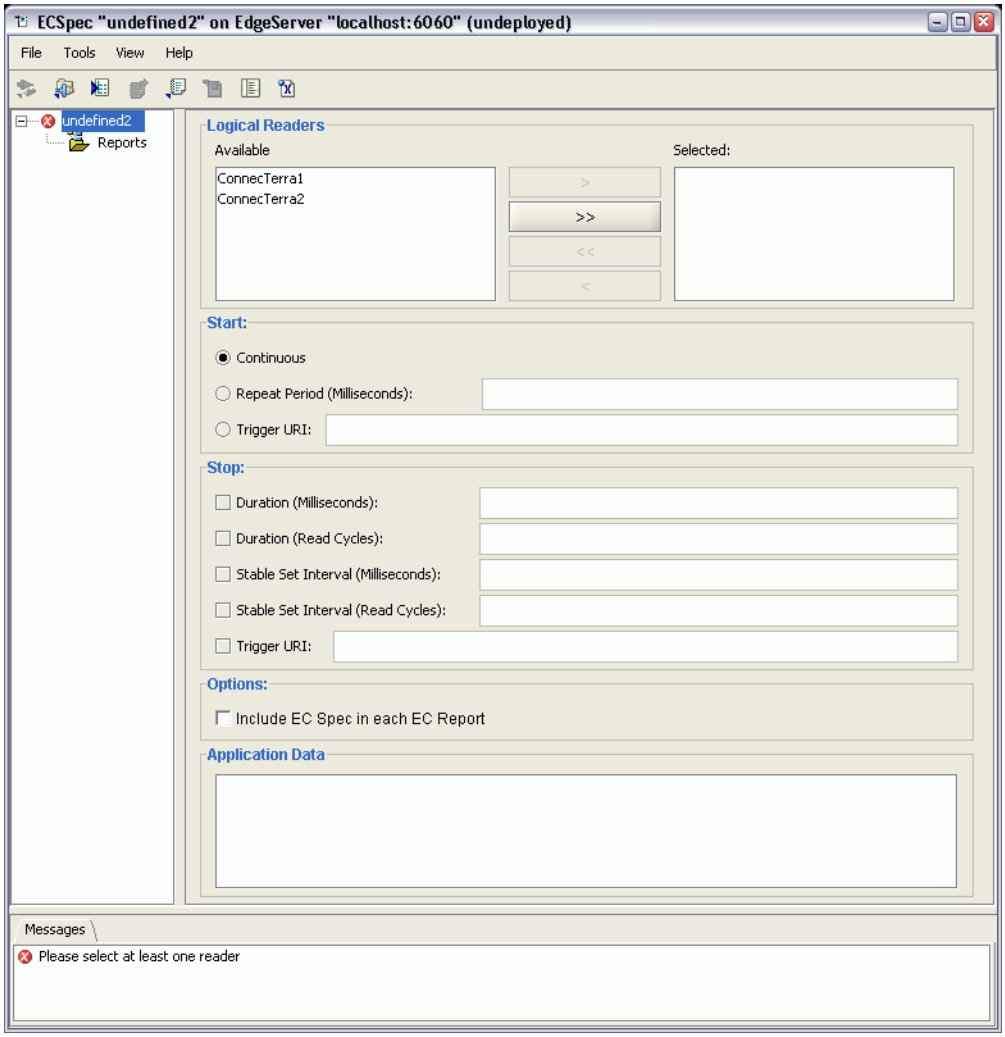
1. Click the New button on the ECSpecs pane.
2. The ECSpec Editor window displays, as shown below.
3. Move one or more logical readers from the Available list to the Selected list using the buttons between the lists.
4. Add a Start condition and one or more Stop conditions to the ECSpec, using the controls in the Start and Stop areas on the ECSpec Editor window. When the ECSpec has an active subscriber, the reader(s) specified in the ECSpec will start reading tags when the start condition is satisfied, and stop reading tags when *any* of the stop conditions are satisfied.

Start Condition	Description
Continuous	Start the next event cycle as soon as the previous event cycle ends.
Repeat Period (Milliseconds)	Start the next event cycle after a specified amount of time has elapsed from the start of the previous event cycle.
Trigger URI	Start the event cycle when a trigger is received.

Stop Condition	Description
Duration (Milliseconds)	Stop reading tags after the stated number of milliseconds.
Duration (Read Cycles)	Stop reading tags after the stated number of read cycles.
Stable Set Interval (Milliseconds)	Stop reading tags after the stated number of milliseconds have elapsed with no new tags seen.
Stable Set Interval (Read Cycles)	Stop reading tags after the stated number of read cycles have elapsed with no new tags seen.
Trigger URI	Stop reading tags when a trigger condition is received.

5. Enter application-specific data in the Application Data field, if any. Data entered here is copied to every report generated from this ECSpec.
6. Add at least one report to the ECSpec. See Creating an ECSpec Report on page 3-18 for more information.
7. Deploy the finished ECSpec by clicking the Deploy As button on the ECSpec Editor toolbar, filling in a name for the ECSpec when prompted, and clicking the OK button to save your changes. You may want to test the ECSpec before deploying. See Testing an ECSpec on page 3-17 for more information.

ECSpec Editor Window (New ECSpec)



Importing an ECSpec

To import an ECSpec:

1. Click the Import button on the ECSpecs pane.
2. Choose the file's location when prompted and click the Open button to import the file.

3. The ECSpec Editor window displays the imported ECSpec.
4. Edit the ECSpec, if desired, and add, remove or edit its report information.
5. Deploy the ECSpec by clicking the Deploy As button on the ECSpec Editor toolbar, filling in a name for the ECSpec when prompted, and clicking the OK button to save your changes. You may want to test the ECSpec before deploying. See Testing an ECSpec on page 3-17 for more information.

Editing an ECSpec

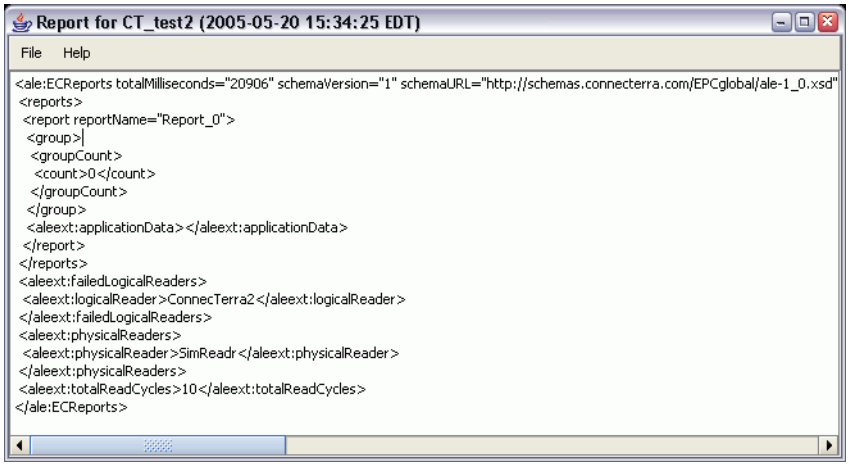
To edit an existing ECSpec:

1. Click an ECSpec on the ECSpecs pane.
2. Click the Edit button.
3. Edit the ECSpec, and add, remove or edit reports and filter information.
4. Deploy the finished ECSpec by clicking the Deploy button on the ECSpec Editor toolbar. You may want to test the ECSpec before deploying. See Testing an ECSpec on page 3-17 for more information.

Testing an ECSpec

To test an ECSpec, click the Test button on the ECSpec Editor toolbar to generate a test report. If the test is successful, a Report window will display XML report data from the Edge Server after the ECSpec has completed one activation (i.e., the Start and Stop conditions have been met).

Report Window



If the report does not appear quickly, a progress dialog will appear. You can cancel the report from this dialog, if desired.

Note: Testing an ECSpec will turn on any readers used by that ECSpec.

The Report window contains menu options to export the report to a file (File | Export...), close the window (File | Close), and access online help (Help | EC Report Help).

Exporting an ECSpec

To export an ECSpec to an XML file:

1. Click an ECSpec on the ECSpecs pane.
2. Click the Edit button on the ECSpecs pane. The ECSpec Editor window displays.
3. Click the Export button on the toolbar (or use the File | Export menu command) to export the ECSpec to an XML file.
4. Choose a path and file from the resulting file dialog, and click Save to save the exported file.

Creating an ECSpec Report

Every ECSpec contains specifications for one or more reports. A report is based on the tags that were detected by the specified readers during the event cycle. It may have more or less information, based on the conditions of the report specification and any report filters or groups that have been defined.

When an event cycle completes, a set of reports for an ECSpec (i.e., an `ECReports` object) is generated and may be sent to subscribers or other clients. For more information on subscribers, see Creating and Removing Subscribers on page 3-11.

> **Note:** An ECSpec must contain at least one report before being deployed.

To add a report to an ECSpec:

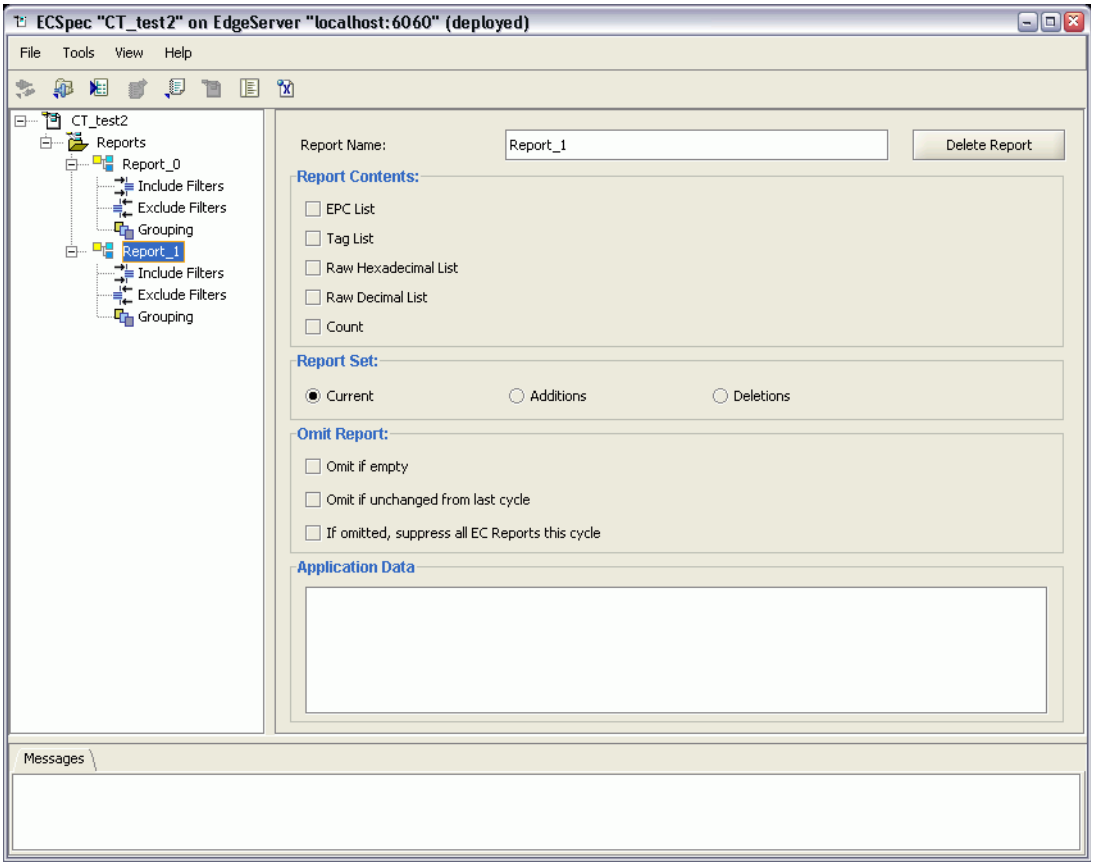
1. On the ECSpec editor window (shown below), click the Reports node on the tree in the left pane. The right pane displays a list of report specifications for the current ECSpec.
2. Click the New Report button in the right pane. The new report is added to the browser tree, and the right pane displays the report properties.

   Each report you add to the browser tree corresponds to an `ECReportSpec` object. For more information on this object, see Chapter 5 of the *Programmer Guide*.
3. Click the Edit button.
4. Enter a unique report name, and choose the report contents using the checkboxes under Report Contents.
5. Choose the report set (i.e., the set of tags to include in the report) using the radio buttons under Report Set.
6. Define the behavior when the report is empty using the checkboxes under Omit Report. If you choose the checkbox labeled *If omitted, suppress all ECReports this cycle*, the existence of this

report will control whether any reports are sent to subscribers. Specifically, if this report is omitted due to one of the other conditions (empty, or unchanged from last cycle), no reports will be sent for the current cycle.

7. Enter application-specific data in the Application Data field, if any. Data entered here is copied to every report generated from this ECSpec.
8. Add one or more filters to the report, if desired. See Creating a Report Filter (below) for more information.
9. The report remains visible in the ECSpec Editor window. Changes are saved when the ECSpec is deployed.
10. One ECSpec can contain many reports. To add more reports to the current ECSpec, simply repeat the steps above. When you have finished editing the ECSpec, deploy it as described in Creating an ECSpec on page 3-15 or Editing an ECSpec on page 3-17.

ECSpec Editor (New Report)



Editing an ECSpec Report

To edit an ECSpec report:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Click an individual report node. The right pane displays a set of properties for the current report. Make any desired changes here.
3. The report remains visible in the ECSpec Editor window. Changes are saved when the ECSpec is deployed.
4. To edit more reports, simply repeat the steps above. Changes are saved when the ECSpec is deployed.

Removing an ECSpec Report

To remove an ECSpec report:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Click an individual report node. The right pane displays a set of properties for the current report.
3. Click the Delete Report button to remove the report. Changes are saved when the ECSpec is deployed.

Creating a Report Filter

Report filters allow you to specify which tags will be included in the report and which tags will be excluded from the report. You can filter on any of the fields for the tag format chosen.

If there are any Include filters, tags that match at least one include filter pattern will be included in the report (unless they also match an Exclude filter). If there are no Include filters, then tags will be included unless they match any Exclude filters.

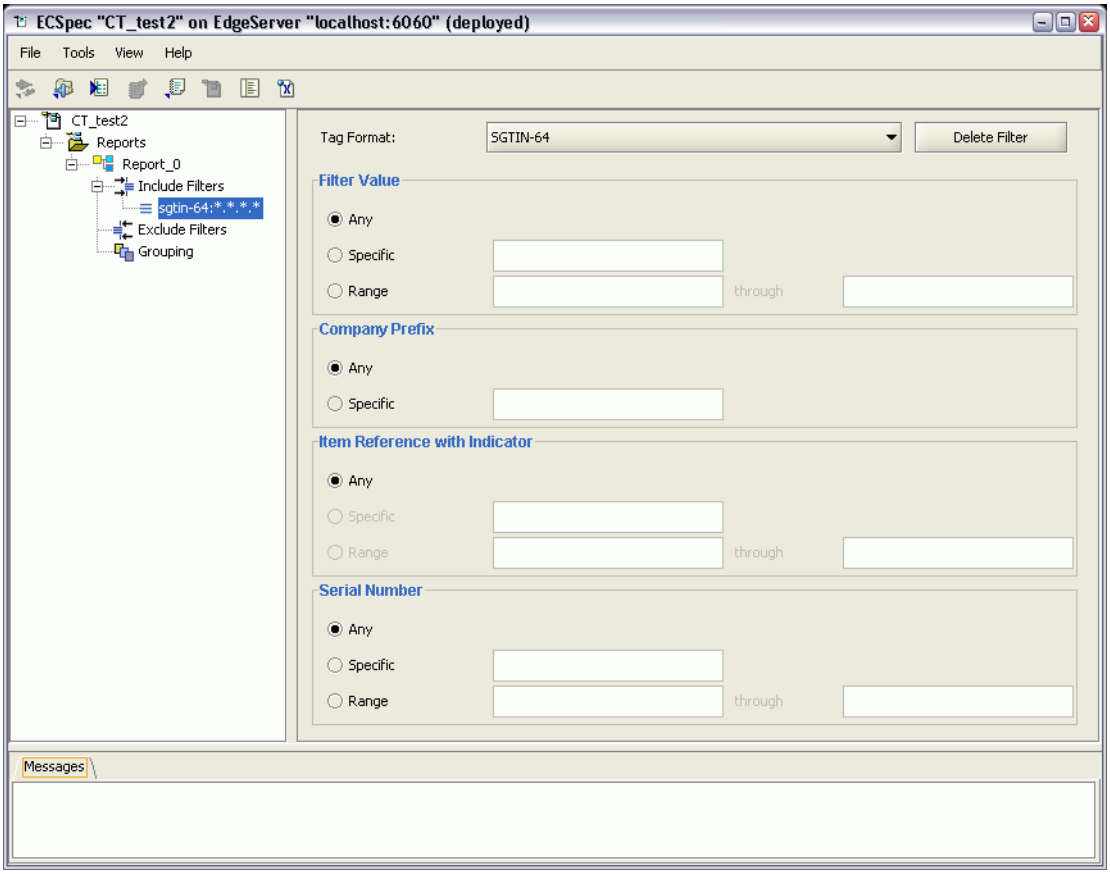
The Include and Exclude filters together make up an `ECFilterSpec` object. For more information on this object, see Chapter 5 of the *Programmer Guide*.

To add a report filter to an ECSpec report:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has two filter nodes beneath it (Include and Exclude). Click a node to display a list of filters of that type in the right pane.

3. Click the New Filter button in the right pane. The new filter is added to the browser tree, and the right pane displays the filter properties.

ECSpec Editor (New Report Filter)



4. Choose a tag format from the drop-down list.
5. Fill in the fields for the areas that display beneath the tag format (the areas shown depend on the format chosen).

   For more information on tag formats, see the *EPCGlobal Tag Data Standards*.
6. The filter remains visible in the ECSpec Editor window until you choose a different node in the tree in the left pane. To add more report filters to the current ECSpec, simply repeat the steps above. Changes are saved when the ECSpec is deployed.

Editing a Report Filter

To edit a report filter:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has two filter nodes beneath it (Include and Exclude).

3. Expand a filter node and click a filter to display the filter properties in the right pane.
4. Edit the fields as desired.
5. The filter remains visible in the ECSpec Editor window until you choose a different node in the tree in the left pane. To edit more report filters in the current ECSpec, simply repeat the steps above. Changes are saved when the ECSpec is deployed.

Removing a Report Filter

To remove a report filter:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has two filter nodes beneath it (Include and Exclude).
3. Expand a filter node and click a filter to display the filter properties in the right pane.
4. Click the Delete Filter button to remove the report filter. Changes are saved when the ECSpec is deployed.

Creating a Report Group

Report grouping allows you to specify how a set of tag data will be separated into subdivisions within a report. You can group on any of the fields for the tag format chosen, and define a series of rules that will generate groups within a report. Both list and count data included in the report are grouped.

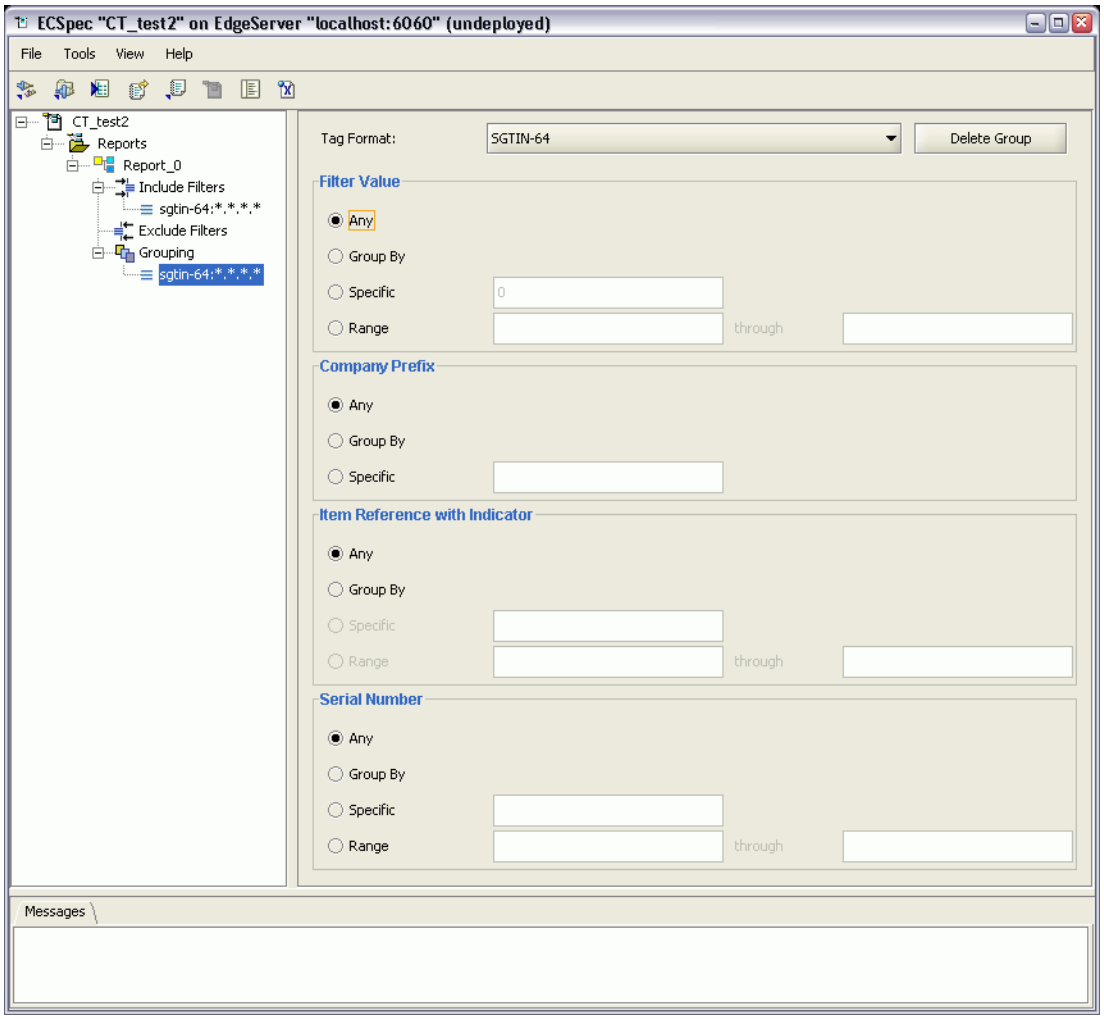
When you create report groups, every tag reported in an event cycle will be part of exactly one group. If a tag does not match any of the pattern URIs in the pattern list, it is included in a special "default group." As a special case of the above rule, if the pattern list is empty (or if no groups are defined), then all tags will be part of the default group. For more information on EPC patterns, see the *Programmer Guide*.

The report group information (all the groups specified under a Grouping node on the left side of the ECSpec editor window) makes up an `ECGroupSpec` object. For more information on this object, see Chapter 5 of the *Programmer Guide*.

To add a report group to an ECSpec report:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has a Grouping node beneath it. Click a node to display a list of filters of that type in the right pane.
3. Click the New Group button in the right pane. The new group is added to the browser tree, and the right pane displays the group properties.

ECSpec Editor (New Report Group)



4. Choose a tag format from the drop-down list.
5. Fill in the fields for the areas that display beneath the tag format (the areas shown depend on the format chosen). For more information on tag formats, see the *EPCGlobal Tag Data Standards*.

   For each area, you can specify grouping behavior:

   - Any – All values belong to a single group (i.e., will appear on a single report).
   - Group By – Create a different group for each distinct value (i.e., generate a separate report for each value).
   - Specific – Create a group for the value specified (and a default group for all other values).
   - Range – Create a group for the range of values specified (and a default group for all other values).

   Fields are inactive when a grouping behavior that is disallowed for a particular area

6. The group remains visible in the ECSpec Editor window until you choose a different node in the tree in the left pane. To add more report groups to the current ECSpec, simply repeat the steps above. Changes are saved when the ECSpec is deployed.

Editing a Report Group

To edit a report group:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has a Grouping node beneath it.
3. Expand the Grouping node and click a group to display the group properties in the right pane.
4. Edit the fields as desired.
5. The group remains visible in the ECSpec Editor window until you choose a different node in the tree in the left pane. To edit more report groups in the current ECSpec, simply repeat the steps above. Changes are saved when the ECSpec is deployed.

Removing a Report Group

To remove a report group:

1. On the ECSpec editor window, expand the Reports node on the tree in the left pane. Each report for this ECSpec is listed beneath the Reports node.
2. Expand an individual report node. Each report node has a Grouping node beneath it.
3. Expand the Grouping node and click a group to display the group properties in the right pane.
4. Click the Delete Group button to remove the report group. Changes are saved when the ECSpec is deployed.

RFID Devices

You can configure RFTagAware to communicate with supported readers in one of two ways:

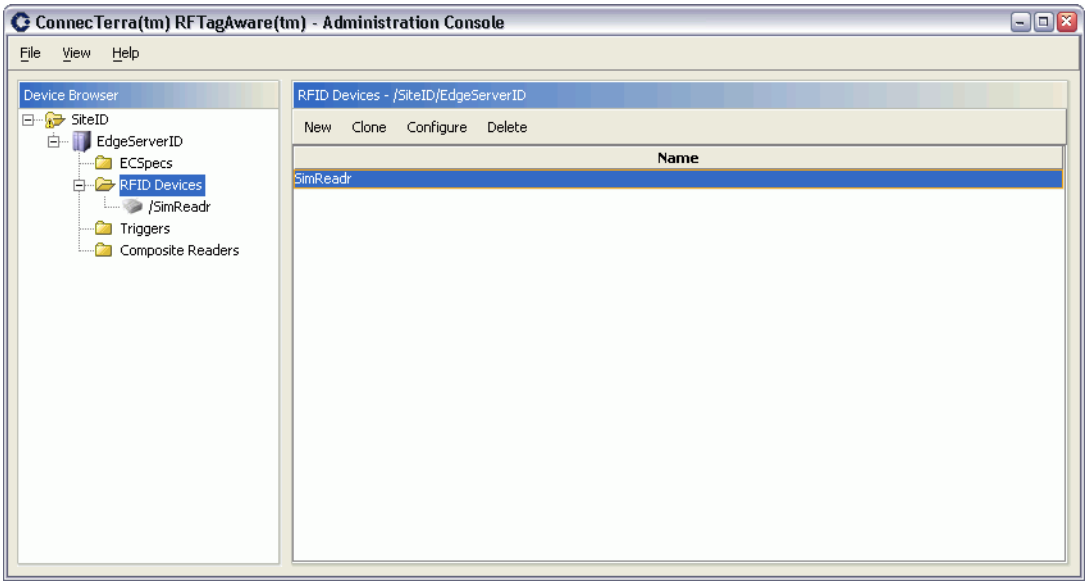
- Editing the reader configuration information using RFID Devices pane on the Administration Console. This is the default for new RFTagAware 1.3 installations. See Setting Up Readers on page 2-21 for complete configuration information.
- Editing the `edge.props` file directly to configure these devices. This is the default for RFTagAware installations prior to version 1.3.

Note: Please be aware that these two methods of configuring supported readers are mutually exclusive. Readers defined using the Administration Console do not appear when editing the `edge.props` file, and vice versa.

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, locate the name of an Edge Server. Open the node for that Edge Server.
3. Click the RFID Devices folder associated with that Edge Server.

RFID Devices Pane



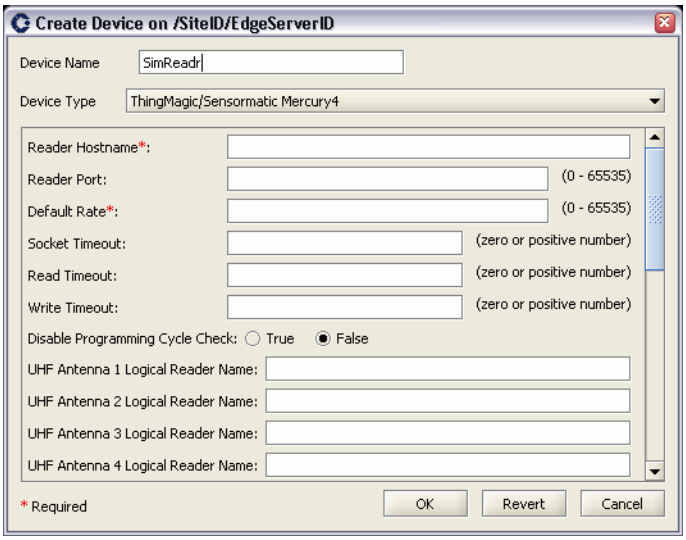
Using this pane, you can add, remove, and configure readers for the Edge Server you selected. Note that if you need to add transient tag filters, you will need to do so by editing the `edge.props` file, as described in Creating Transient Filters on page 3-27.

Adding and Removing Readers

To add a new physical reader to the selected Edge Server:

1. Click the New button on the RFID Devices pane.
2. The Reader Configuration dialog displays. Enter a device name (which will appear in the Administration Console device browser) and choose a device type to display reader-specific properties.

Reader Configuration Window (New Reader)



3. Fill in values for the reader-specific properties shown. Properties that must be filled in will display an asterisk (*) to the right of the field. You may wish to consult the *RFTagAware Reader Configuration Guide* or manufacturer's manual for your reader for more information on these properties and their allowable values.
4. Click the OK button to save your changes and add the new reader to the Edge Server.

To make a copy of an existing reader definition:

1. Click a reader from the list on the on the RFID Devices pane.
2. Click the Clone button.
3. The Reader Configuration dialog displays, displaying the duplicated device definition and properties from the item you originally clicked.
4. Edit existing device name, and change the values of the device-specific properties as needed.
5. Click the OK button to save your changes and add the new device to the Edge Server.

To delete a reader from the selected Edge Server:

1. Click a reader from the list on the on the RFID Devices pane.
2. Click the Delete button.
3. Confirm that you want to delete this device from the Edge Server.

Configuring Readers

To configure an existing reader:

1. Click a reader from the list on the on the RFID Devices pane.
2. Click the Configure button.
3. The Reader Configuration dialog displays, displaying the device name, device type, and properties.
4. Make configuration changes as necessary, and click the OK button to save the changed reader configuration to the Edge Server.

Creating Transient Filters

Transient filters allow you to "smooth out" the stream of raw data coming from the reader. You can apply a transient filter to any reader antenna (i.e., logical reader). Different logical readers may share the same filter settings, or have different settings.

For each transient filter you add, these three parameters control its operation:

- `minReads` – The number of times a tag must be read before being included in the filter (i.e., visible to the event cycle).
- `firmInterval` – The maximum time (in milliseconds) allowed between reads that increase the `minReads` count.
- `expiredInterval` – The maximum duration (in milliseconds) for a tag not to be read before expiring from the filter.

To create a transient filter, edit the `edge.props` file as described below. First, add the following lines to define a named filter (altering the parameter values as desired):

```
com.connecterra.ale.filter.<filter_name>.class=
   com.connecterra.ale.filtertypes.TransientFilterFactory
com.connecterra.ale.filter.<filter_name>.minReads = 3
com.connecterra.ale.filter.<filter_name>.firmInterval = 1400
com.connecterra.ale.filter.<filter_name>.expiredInterval = 1400
```

Then, for each logical reader to which you want to add the filter, add a line like the one below to the `edge.props` file. Here, *<reader_name>* is the device name of a reader, and *<filter_name>* is the name of the filter.

```
com.connecterra.ale.logicalReader.<reader_name>.filters = <filter_name>
```

To apply the same filter to more than one logical reader, you may specify the same filter name for more than one reader. Even though more than one logical reader refers to the same filter name, each logical reader is processed by a different filter instance.

Monitoring Readers

You can monitor how long read cycles are taking, and keep track of how many read cycles have occurred since you started the Edge Server. You can see how many tags each reader is reading.

You can also examine configuration information for each reader, and examine the messages and warnings that the reader generated.

For details, see:

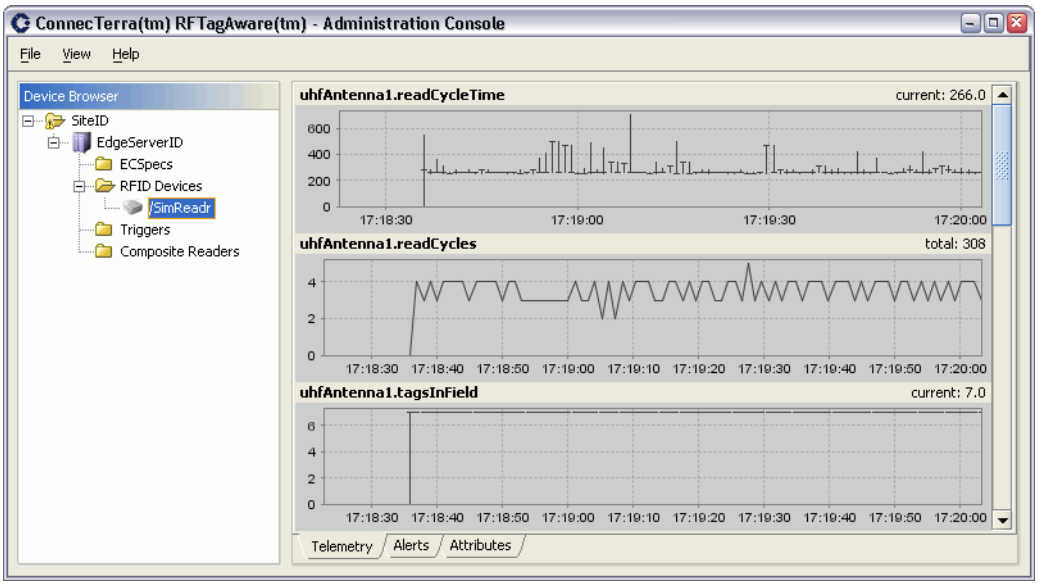
- Reader Telemetry (page 3-28)
- Reader Alerts (page 3-29)
- Reader Attributes (page 3-30)

Reader Telemetry

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, locate the name of an Edge Server. Open the node for that Edge Server.
3. Open the Readers folder associated with that Edge Server.
4. Click the name of a physical reader.
5. In the right panel, click **Telemetry**. Telemetry information for each logical reader/antenna appears.

Reader Telemetry Tab



readCycles

Shows when read cycles occurred.

total	The total number of read cycles completed since the Edge server was started.
horizontal axis	Time stamps, in 24 hour time.
vertical axis	Shows how many read cycles were completed within the telemetry interval configured in the Edge Server. When the telemetry interval is very short (as the default 250ms is), this value is likely to be either zero (no read cycles) or one (one read cycle). If the telemetry interval is longer, larger values will be displayed when many read cycles are completed within one telemetry interval.

readCycleTime

Shows how much time each reader is taking for a read cycle.

current	The value from the most recent read cycle.
horizontal axis	Time stamps, in 24 hour time.
vertical axis	Minimum and maximum read cycle times, in milliseconds.

tagsInField

Shows how many tags this reader read in the field.

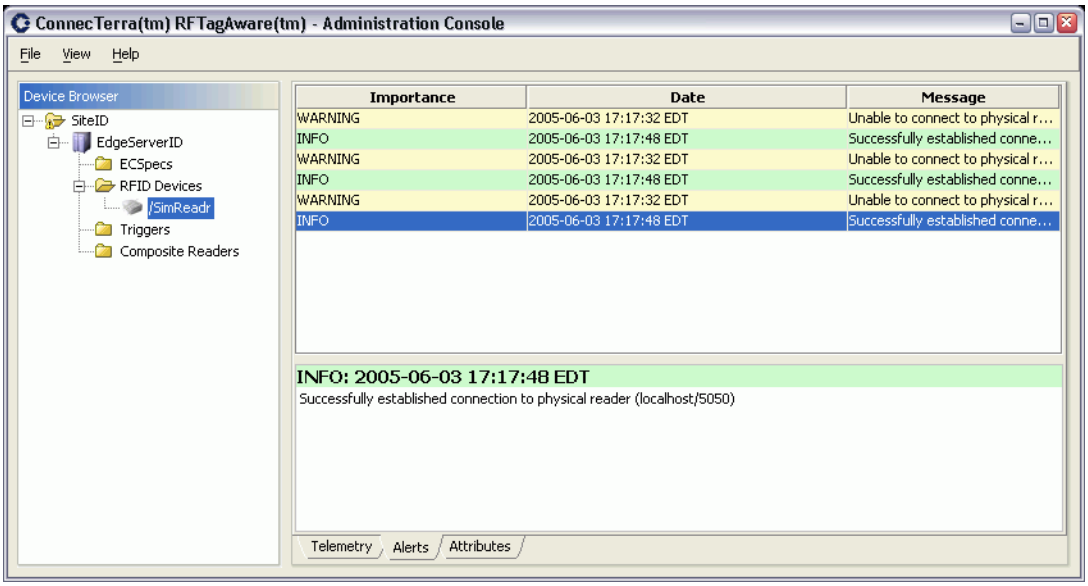
current	The value from the most recent read cycle.
horizontal axis	Time stamps, in 24 hour time.
vertical axis	Shows how many tags this reader read during each read cycle.

Reader Alerts

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, locate the name of an Edge Server. Open the node for that Edge Server.
3. Open the Readers folder associated with that Edge Server.
4. Click the name of a reader, and n the right panel, click the **Alerts** tab.

Reader Alerts Tab



The reader alerts tab shows messages and warnings generated by the reader

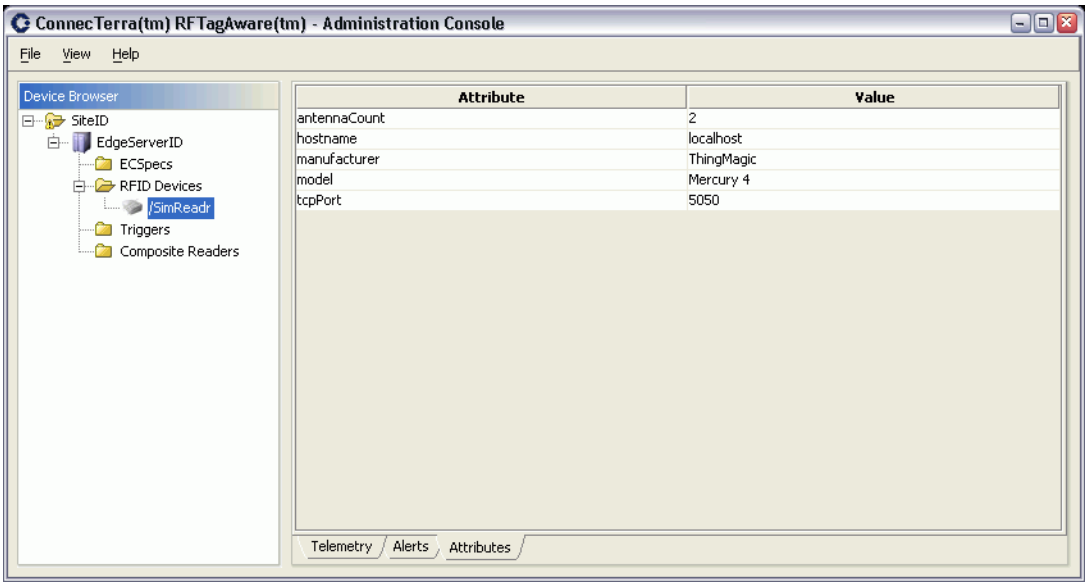
Importance	Can be: INFO (green) WARNING (yellow) SEVERE (red)
Date	The date and time the reader generated the message.
Message	The text of the message.

Reader Attributes

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, locate the name of an Edge Server. Open the node for that Edge Server.
3. Open the Readers folder associated with that Edge Server.
4. Click the name of a reader.
5. In the right panel, click **Attributes**.

Reader Attributes Tab



The reader attributes tab shows configuration information for this reader. The reader sends this configuration information to the Edge Server.

antennaCount	How many antennas this reader has.
hostname	The hostname or IP address that the Edge Server is using to communicate with this reader.
manufacturer	The reader's manufacturer.
model	The reader's model.
tcpPort	The TCP port that the Edge Server is using to communicate with this reader.

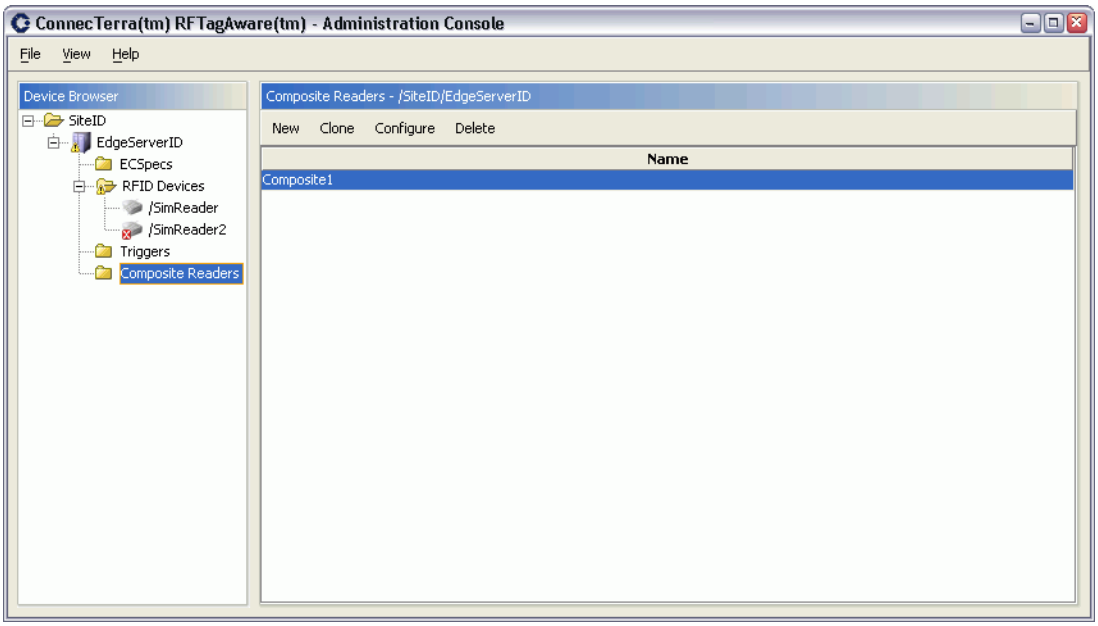
Composite Readers

Composite readers are created by combining existing logical readers. They provide an easy way to add or remove logical readers when one or more ECSpecs have already been defined. For more information on composite readers, see Using Composite Readers on page 2-23.

How to display:

1. Start the Administration Console, as described in Starting and Stopping RFTagAware on page 2-36.
2. In the left panel, locate the name of an Edge Server. Open the node for that Edge Server.
3. Click the Composite Readers folder associated with that Edge Server.

Composite Readers Pane



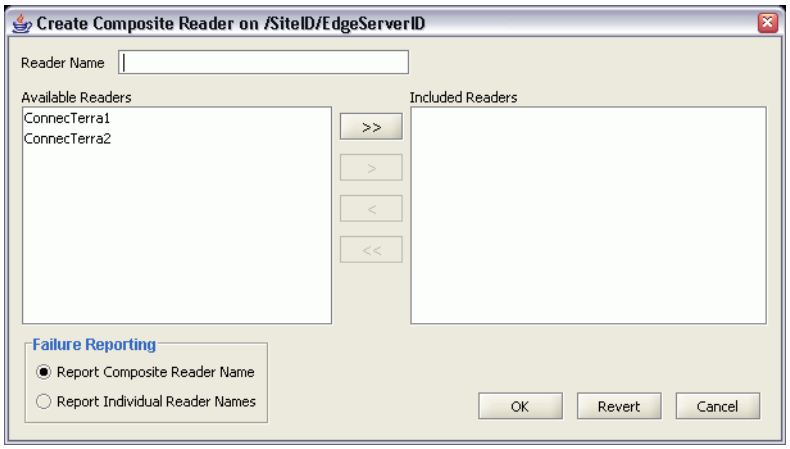
Using this pane, you can add, remove, and configure Composite Readers.

Adding and Removing Composite Readers

To add a new composite reader to the selected Edge Server:

1. Click the New button on the Composite Readers pane.
2. The Composite Reader Configuration window displays.

Composite Reader Configuration Window (New Reader)



3. Fill in a reader name (which will appear in the Administration Console device browser and in the list of readers shown in the ECSpec Editor window), and move one or more logical readers from the Available Readers list to the Included Readers list using the arrow controls between the lists.
4. Click the OK button to save your changes and add the new composite reader to the Edge Server.

To make a copy of an existing composite reader:

1. Click a composite reader from the list on the Composite Readers pane.
2. Click the Clone button. The Composite Reader Configuration dialog displays the reader information.
3. Fill in a new composite reader name, and change the included readers as needed.
4. Click the OK button to save your changes and add the cloned composite reader to the Edge Server.

To delete a reader from the selected Edge Server:

1. Click a composite reader from the list on the on the Composite Readers pane.
2. Click the Delete button.
3. Confirm that you want to delete this composite reader from the Edge Server.

Configuring Composite Readers

To configure an existing composite reader:

1. Click a composite reader from the list on the on the Composite Readers pane.
2. Click the Configure button.
3. The Composite Reader Configuration window displays, displaying the reader name and list of included readers.
4. Make configuration changes as necessary, and click the OK button to save the changed composite reader configuration to the Edge Server.

Triggers

Applications may define event cycle specifications (`ECSpec`) and programming cycle specifications (`PCSpec`) where the beginning and/or end of each cycle is triggered by external events. Triggers are supported for custom deployments. Contact your ConnecTerra support representative for more information.

Chapter 4: Using the Reader Simulator

Contents

This chapter describes how to use the RFTagAware Reader Simulator.

Overview

Your RFTagAware installation includes reader simulator software that you can use for software evaluation, application development, and testing. This is a Java application that provides a minimal simulation of a ThingMagic Mercury4 reader or (with minor configuration) a Printronix printer.

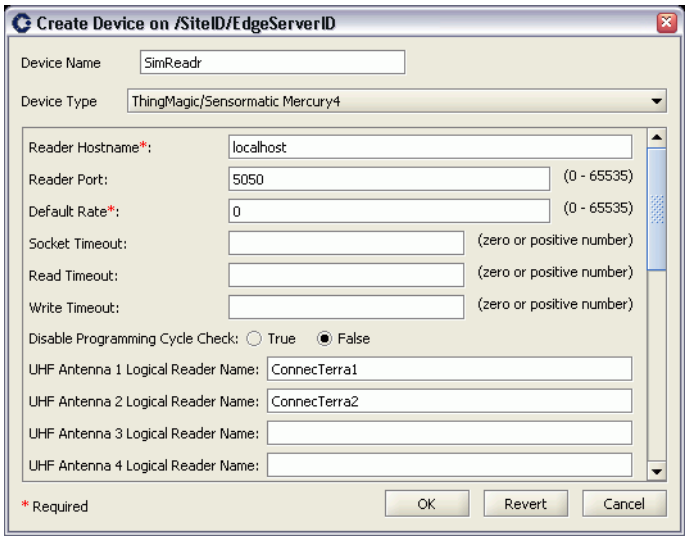
The default `edge.props` file that is initially installed on your system contains the following block of properties definitions, which defines a ThingMagic Mercury4 reader with two antennas running on your local system on port 5050. This set of properties refers to the RFTagAware Reader Simulator.

```
com.connecterra.ale.reader.SimReadr.class =
   com.connecterra.ale.readertypes.ThingMagicMercury4PhysicalReader
com.connecterra.ale.reader.SimReadr.hostname = localhost
com.connecterra.ale.reader.SimReadr.port = 5050
com.connecterra.ale.reader.SimReadr.defaultRate = 0
com.connecterra.ale.reader.SimReadr.uhf2LogicalReaderName = ConnecTerra2
com.connecterra.ale.reader.SimReadr.uhf1LogicalReaderName = ConnecTerra1
```

Note that a single `edge.props` file may contain properties definitions for many readers. For testing and development purposes, you may want to keep some definitions (for example, for the simulator) in the file but inactive. You can prevent RFTagAware from trying to communicate with a reader by simply commenting out the first line of a reader's property definitions. (To comment out a line, place a # at the beginning of the line.)

If you choose to configure the Reader Simulator through the RFID Devices pane on the Administration Console instead of using `edge.props`, you will need to enter properties into the Reader Configuration dialog as shown below. See RFID Devices on page 3-24 for full instructions on configuring readers.

Reader Configuration for the Reader Simulator.



The compiled sample applications will work "out-of-the-box" with the pre-configured Reader Simulator. (RFTagAware sample applications are described in the *RFTagAware Programmer Guide*.)

Configuring the Reader Simulator

You can configure the Reader Simulator by editing the `RunReaderSim` script in your RFTagAware `bin` directory.

As long as you plan to run the simulator on the system where RFTagAware is installed, you can configure the simulator simply by editing this script. If you want to run the simulator on a different machine, you will need to make sure the simulator is launched by a similar command that finds the local JVM and the RFTagAware classes.

Command Line Switches and Arguments

Command line switches and arguments to the `RunReaderSim` script are contained in the line:

```
"%JAVA_HOME%\bin\java" -jar "%RFTA_ROOT%\lib\readersim.jar" -epcIndexTableURL
-printer -printerPort 9101 "%EPCINDEXTABLEURL%" %* %READER_PORT% %TAG_COUNT%
%ANTENNA_COUNT%
```

This script is preconfigured at installation to use your system's JVM and to point to the RFTagAware `lib` directory to find the Java classes needed by the simulator (the `classpath` parameter).

The `printer` parameter indicates that you are simulating an RFID printer, in addition to a reader. You must configure a Printronix printer in RFTagAware as well as including this command line switch. The `printerPort` parameter specifies a port that will receive the data to be printed. The port defaults to 9100 if omitted.

The `epcIndexTableURL` parameter specifies the location of the EPC Company Prefix Index translation table. RFTagAware uses this table to decode tags encoded in certain 64-bit formats, as defined by the *EPCglobal Tag Data Standards*. By default, `RunReaderSim` attempts to connect to the EPCglobal web site (http://onsepc.com/ManagerTranslation.xml) to find this table:

```
-epcIndexTableURL "%EPCINDEXTABLEURL%"
```

If you are running the Reader Simulator in an environment that is not connected to the Internet, you can use the local copy (installed in the etc directory) instead. Edit the `RunReaderSim` file to point to the local file, using the `file:` syntax shown below:

```
-epcIndexTableURL "file:ManagerTranslation.xml"
```

If the pathname contains spaces, replace the spaces with %20, for example:

```
file:///C:/Program%20Files/Connecterra...
```

> **Note:** If you use a local file, it is up to you to keep it up to date by copying down subsequent versions from the EPCglobal site as necessary.

The last three command line arguments for the Reader Simulator are, in order, the port on which the simulator will communicate, the maximum number of tags to be seen by each antenna, and the number of antennas.

In the example command from the preinstalled `RunReaderSim` script these values are set to:

- port - 5050
- number of tags - 7
- number of antennas - 2

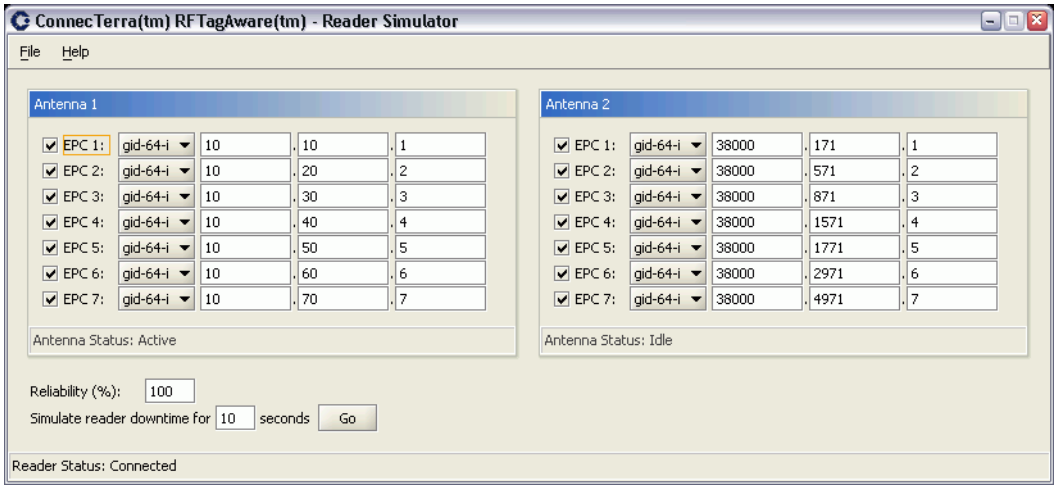
These example values are not the defaults. If you do not specify these three arguments, the port will be set to 8080, the number of tags to 6, and the number of antennas to 2. Note that the ThingMagic Mercury3 has only 2 antennas, but the Mercury4 has up to 16 antennas.

Starting the Reader Simulator

To start the Reader Simulator, go to the `/bin` directory and run the script:

```
RunReaderSim
```

The Reader Simulator appears:



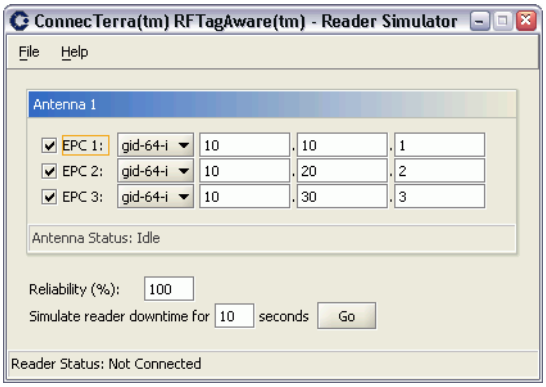
Each field is described in Reader Simulator Reference on page 4-7. For now, just note that:

- Using the command line arguments in the `RunReaderSim` file, this simulated reader has two antennas, each of which can see seven tags.
- The reader status message at the bottom of the window indicates that the reader is not connected. In this example, we haven't started the Edge Server yet, so the reader has nothing to connect to. This will automatically change when we start the Edge Server, as described in Starting the Edge Server on page 4-5.
- You can change the number of antennas and tags that the simulated reader sees, by editing the `RunReaderSim` file. Note that the ThingMagic Mercury3 reader has a maximum of two antennas, but the ThingMagic Mercury4 reader has a maximum of 16 antennas.

The `RunReaderSim` script specifies the number of tags each antenna should see (7) and the number of antennas the Reader Simulator should have (2):

```
set TAG_COUNT=7
set ANTENNA_COUNT=2
```

Try changing these values to "3" and "1", respectively. Stop the Reader Simulator, then restart it and you will see that it now has only one antenna, which sees only three tags:



Starting the Edge Server

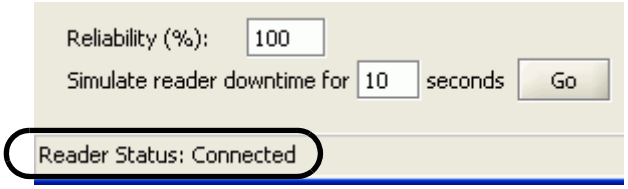
To start the Edge Server, go to the `/bin` directory and run the script:

```
RunEdgeServer
```

The Edge Server uses the reader configuration properties definitions in the `edge.props` file to connect with the Reader Simulator. The initial installed version of `edge.props` is preset to connect to the simulator on `localhost:5050`, which coincides with the command line argument in the initial `RunReaderSim` script. (If you have changed the initial installed `edge.props` and you now want to run the simulator, see Setting Up Readers on page 2-21 for information about the reader properties you need to connect with the simulator.)

If you start the Edge Server when the simulator is already running, the Edge Server will connect immediately. If you start the simulator after the Edge Server, the Edge Server will connect to the simulator whenever an application attempts to connect to the simulator, for instance when you run the `RunQuickTest` script.

If you take a look at the Reader Simulator once the Edge Server has connected to it, you will see that its status has changed to `Connected`.



Using the Reader Simulator with the QuickTest Utility

This section shows you how to use the Reader Simulator in conjunction with the QuickTest utility. The QuickTest utility provides a quick test to see that your Edge Server and readers are operating correctly. Observing the interactions of the QuickTest utility and the Reader Simulator is a good way to get familiar with some basic RFTagAware features.

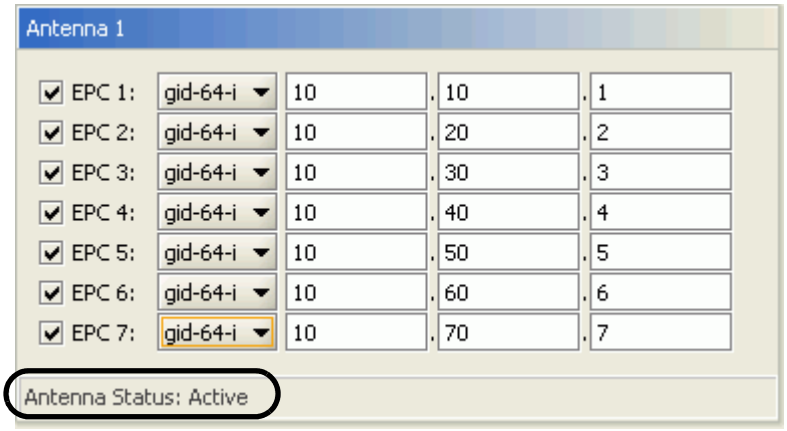
1. First, set up your desktop so you can see both the Reader Simulator and a new console window at the same time.

2. In the new console window, start the QuickTest utility by going to the `bin` directory and running the script:
   ```
   RunQuickTest
   ```

   In the console window, you can see that QuickTest first connects to the Edge Server, then starts testing a logical reader called `ConnecTerra1`:
   ```
   Connecting to edge server...
   Finished connecting to edge server.
   Testing logical reader ConnecTerra1...
   ```

   `ConnecTerra1` is the logical reader name of the first antenna on the Reader Simulator — here is the relevant line from the default `edge.props`:
   ```
   com.connecterra.ale.reader.SimReadr.uhf1LogicalReaderName = ConnecTerra1
   ```

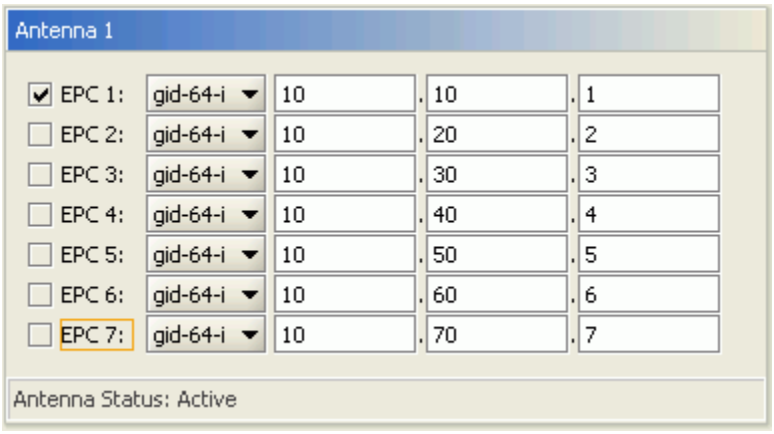
   Within the Reader Simulator, `ConnecTerra1` is called Antenna 1, so when QuickTest starts testing `ConnecTerra1`, the Reader Simulator shows Antenna 1's status turning to `Active`:

   Now take a look at the tags that QuickTest read from `ConnecTerra1`:
   ```
   Logical reader ConnecTerra1 read the following 7 EPCs:
   urn:epc:tag:gid-64-i:10.50.5
   urn:epc:tag:gid-64-i:10.40.4
   urn:epc:tag:gid-64-i:10.10.1
   urn:epc:tag:gid-64-i:10.30.3
   urn:epc:tag:gid-64-i:10.70.7
   urn:epc:tag:gid-64-i:10.20.2
   urn:epc:tag:gid-64-i:10.60.6
   Finished testing logical reader ConnecTerra1.
   ```

   As you can see, these are the same EPC values for the seven tags that Antenna 1 in the Reader Simulator is seeing (shown in the image above).

3. Now try changing what Antenna 1 of the Reader Simulator is seeing — uncheck all but the first EPC value:

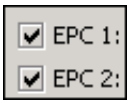


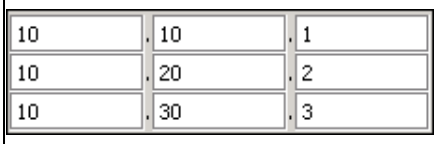
4. Run QuickTest again. This time it reports seeing only one tag for Antenna 1.

```
Testing logical reader ConnecTerra1...
Logical reader ConnecTerra1 read the following EPC:
urn:epc:tag:gid-64-i:10.10.1
Finished testing logical reader ConnecTerra1.
```

Reader Simulator Reference

Menus	
File	Lets you exit the Reader Simulator.
Help	Displays copyright and version information.

Antenna Pane	
EPC 1:, EPC2: etc. check boxes 	Check each tag you want the reader to see. Uncheck the tags you do not want the reader to see.
Tag format drop down menu 	Lets you specify the formats of the tags that the Reader Simulator is seeing. See Using 64-bit Tags on page 2-25 and Using 96-bit Tags on page 2-26 for more information on tag formats.

Antenna Pane	
EPC numerical value fields  	Shows the numerical value of the EPC.
Antenna Status	Can be Active or Idle. An antenna is Active when the Edge Server is asking it to do something (for example, when a program has invoked ALE API methods such as `immediate`, `poll`, and `subscribe`).
Reliability (%)	Often, tag readers do not see all the tags in their fields. You can use this field to simulate various levels of reader reliability for testing.
Simulate reader downtime	Lets you simulate a reader being unavailable for a specified number of seconds, then becoming available again. Used for testing.
Reader Status	Can be Connected or Not Connected. A reader is Connected if it is in communication with the Edge Server.

Index

L

M

N

P

Q

R

S

T

U

V

W