



System Administrator's Guide

Copyright © 2002, Synchron Networks, Inc. All rights reserved. No part of this documentation may be reproduced in any form or by any means or used to make any derivative work without written permission from Synchron Networks. Permission to duplicate all or part of this documentation exclusively for internal use by the purchaser of a license for the software described herein is hereby granted for support of the licensed use of the software.

THE SPECIFICATIONS AND INFORMATION REGARDING THE SOFTWARE DESCRIBED IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE, BUT ARE PRESENTED WITHOUT WARRANTY, TERM, OR CONDITION OF ANY KIND, EITHER IMPLIED OR EXPRESSED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES.

The license and limited warranty for the software described in this manual are set forth in the license.txt file supplied with the product CD and are incorporated herein by reference.

Synchron Networks and Everserve are trademarks of Synchron Networks, Inc.

All other company and product names may be trademarks of the respective companies with which they are associated.

This product incorporates FIORANOMQ from Fiorano Software, Inc., and other software provided under license by third parties (see [Third Party Software](#)). Use of such software is subject to the terms and conditions of the corresponding licenses. Electronic copies of those license agreements are included in the Third Party Software directory of the distribution media.

Part No. 2.0-3.20-SAG-03

Table of Contents

<i>Introduction</i>	<i>13</i>
About this Guide	14
Intended Audience	15
Technical Support	15
Typographical Conventions.....	16
Terminology.....	17

Part 1

Everserve Installation and Configuration

<i>Planning Your Everserve Community.....</i>	<i>21</i>
Before You Begin	22
Summary.....	22
Obtaining an Inventory of Devices	23
Designing Your Community	24
Determining Logical Target Groupings.....	24
Using Relays.....	25
Planning for Fail-over and Redundancy.....	26
<i>Installation</i>	<i>27</i>
Pre-installation Checklist.....	28

System Requirements	29
Supported Databases	29
Supported Platforms	29
Community Managers, Publishers, and Relays	30
Targets	30
Setting up the Database	30
Database Security	30
Database Optimization	31
MySQL	32
Installing MySQL on Windows Systems	32
Installing MySQL on Solaris Systems	34
Installing Everserve on Windows Devices	36
Target Only Installations	36
Community Manager, Publisher, and Relay Installations	43
Silent Installations	61
Installing Everserve on Solaris Devices	64
PKG-ADD Installation for Solaris	64
Summary	64
Uninstall Prior Versions of Everserve	64
Installing MySQL	66
Installing Everserve	66
Silent Installations	72
Installing Everserve for Internet Messaging	75
Configuration Summary	76
JMS Server Only Installations	76
Windows JMS Only Installations	77

Solaris JMS Only Installations	83
Post Installation Tasks	85
System Configuration	87
Configuring Alternate TCP and JMS Ports	88
Configuring RMI TCP Port	88
Configuring Fiorano JMS Ports	89
Limiting Target Capabilities	90
Configuring Everserve to Run as a Specific User	91
Locking Down Directories, Files, and Executable Operations	92
Configuring System Logs.....	94
Changing Logging Layout and Format.....	94
Specifying Logging Priority Values	96
Specifying Format and Destination of STDOUT Logging.....	97
Specifying Format and Destination of STDERR Logging.....	97
Logging Java Exceptions	98
Logging Everserve Start and Stop Activity	98
Logging Server Connectivity Problems	99
Setting-up Email Notification.....	100
Setting Delivery Timeouts	102

Part 2

Community Management

<i>Command Line Interface Operations</i>	105
Using the Interactive Command Shell	106
Controlling Operative States	107
Starting Everserve	107
Stopping Everserve	107
Pausing Everserve	108
Resuming Everserve	108
Managing Communities	109
Community Descriptions	109
Creating a New Community	110
Changing a Community Definition	111
Deleting a Community	112
Recreating a Community Seed	113
Managing Peers	114
Peer Descriptions	114
Creating a Peer	115
Adding Peers to a Community	116
Specifying Peer Connections in the Community	117
Connecting a Sender to an Everserve Device	118
Default Peer Connections (When a Sender is Not Specified)	119
Specifying Peer Connections	120
Creating and Adding Several Peers to a Community	122
Changing a Peer Definition	124

Changing the Sender Connections for a Peer	126
Removing a Sender from a Peer	126
Activating a New Everserve Device	128
Removing a Peer from a Community	129
Deleting a Peer	130
Publishing Commands	131
Delivering Packages	131
Executing Files in Batch Mode	133
Information Services Commands	135
Specifying Date Ranges	135
Show Commands	135
Show Communities	136
Show Peers	137
Show Processes	140
Show Receipts	142
Show Deliveries	144
Show Deliverylog	145
List Commands	147
Listxml	147
List Roles	147
Listing Environment Settings	147
Listing Everserve Version	148
Help	148

<i>Creating a Community</i>	149
Community Hierarchy	150
Multiple Community Managers	151
Multiple Publishers	151
Tiered Relays and Multiple Targets	151
Creating the “Example” Community	152
Using the Everserve Command Shell	152
Create the Community	152
Create the Community and Community Managers	152
Create the Publishers	153
Create the Relays	154
Create the Targets	157
Joining the Community	162
Configuring Backup Publishers	164
Creating Backup Communities	166
Publishing Packages	172
Publishing to an Entire Community	172
Publishing to a Relay	175
Publishing to a Single Target	177
Publishing to a Lists of Targets	179
<i>Everweb Operations</i>	181
Connecting to Everweb	182
Everweb Page Overview	184
Page Navigation	185
Status Indicators	185

Managing Communities	186
Creating Communities.....	186
Changing Community Definitions.....	189
Deleting Community Definitions	190
Managing Peers.....	191
Creating Peers	191
Adding Peers to a Community	195
Changing Peer Definitions	199
Removing a Peer from a Community.....	200
Deleting Peer Definitions.....	204
Viewing Community Peers	205
Viewing System Log Files.....	206
Viewing Server Log File	206
Viewing the Processes Log File	207
Server Operations.....	210
Logging Off Everweb	211

Part 3

Maintenance and Troubleshooting

<i>Database and Filestore Maintenance</i>	<i>215</i>
Database Tables	216
Archiving Records.....	233
Best Practices	233
Specifying Date Ranges.....	233
Archiving Delivery Records	234

Archiving Process Records	235
Archiving Deliverylog Records.....	236
Archiving All Records	237
Purging Records.....	238
Best Practices.....	238
Purging Delivery Records.....	239
Purging Deliverylog Records	240
Purging Process Records.....	241
Purging All Records.....	242
Restoring Archived Records.....	243
Restoring Delivery Records.....	243
Restoring Deliverylog Records	244
Restoring Process Records.....	245
Restoring All Records.....	246
Restoring Purged Records	247
Error Recovery	248
Backing-up and Restoring Databases.....	248
System Logs	251
Server Logs.....	252
Auditing the System.....	252
Database Tables	253
Viewing Server Log Files.....	253
Controlling everserve.log Verbosity.....	253
Log4j.xml Properties	254
Log4j Configuration Quick Reference	258

everserve_log.bak File Maintenance	259
Process Logs	259
<i>Appendix A: Summary of CLI Commands</i>	<i>261</i>
<i>Appendix B: Sample Script for Creating Communities.....</i>	<i>265</i>
<i>Appendix C: Third Party Software.....</i>	<i>271</i>
Glossary	273
Index.....	277

Introduction

This *System Administrator's Guide* provides information on many of the fundamental principles used to properly set up community devices to send or receive messages deployed by Everserve. This guide provides systematic instructions on how to install Everserve, create communities, and deliver data and applications to mass numbers of globally distributed Everserve devices.

The topics in this Introduction include:

- [About this Guide](#)
- [Intended Audience](#)
- [Technical Support](#)
- [Typographical Conventions](#)
- [Terminology](#)

About this Guide

This *System Administrator's Guide* describes how to install and configure your Everserve network for optimum secure content and application deployment. This guide provides information and instructions on how to establish and configure a community of Everserve devices. For information on how to create and publish packages to community peers, refer to the *Everserve Publisher's Guide*.

This guide is divided into the following sections:

[Introduction](#), which you are reading now, introduces the structure of this guide, introduces key terms and conventions used in this manual, provides information about additional resources, and describes Everserve's key features.

Part 1 **Everserve Installation and Configuration**

[Planning Your Everserve Community](#) provides information and helpful tips on planning the community hierarchy and transmission routing, pre-installation checklists, and other aspects of the installation process that should be considered before installing Everserve.

[Installation](#) describes how to install a MySQL database, install Everserve onto Windows and Solaris devices, and provides information on how to set up a community with secure Internet messaging.

[System Configuration](#) provides information on how to configure port connections, system logs, set up email notification, and change delivery time-outs.

Part 2 **Community Management**

[Command Line Interface Operations](#) describes how to use the command line interface to create, populate, and configure a community, and describes how to issue commands to community devices.

[Creating a Community](#) describes how to create, populate, and configure a complex community consisting of backup community managers, backup publishers, relays and multiple targets.

[Everweb Operations](#) describes how to create a community and peers using Everserve's graphical user interface, Everweb.

Part 3 **Maintenance and Troubleshooting**

[Database and Filestore Maintenance](#) describes the recommended routine record archiving and purging operations for Everserve devices.

[System Logs](#) provides information on how to monitor system activity and errors using Everserve system log files.

[Appendix A: Summary of CLI Commands](#) lists all commands and corresponding parameters that can be issued from the command line interface.

[Appendix B: Sample Script for Creating Communities](#) provides an example script of all commands used to create the community described in the section [Creating a Community](#).

Appendix C: Third Party Software provides the copyrights to all third party software used in the development of Everserve.

Glossary provides an alphabetical list of terms that are used in this guide.

Intended Audience

This guide describes the concepts, processes, and procedures that a System Administrator should know or have a reasonable understanding of, to efficiently manage communities of Everserve devices. This guide has been written with the assumption that the reader or end user has an advanced familiarity with Windows or UNIX administration, however, no previous knowledge of Everserve is required. Additionally, a familiarity with managing database information is highly recommended.

Technical Support

Technical support is available by:

- Calling (831)247-3983
- Accessing Synchron's Web site at <http://www.synchronnetworks.com/support>
- Sending email to Synchron's technical support experts at support@synchronnetworks.com

You can also access newsgroups that contain discussions and links to related forums about Everserve at:

- <news://newsgroups.synchronnetworks.com>
- <http://www.synchronnetworks.com/newsgroups>

Typographical Conventions

The following typographical and keying conventions are used in this guide:

Convention	Description
Bold	Used to indicate emphasis and to distinguish user entries and mouse clicks.
<i>Bold Italic</i>	Used to identify a <i>new term</i> . All new terms and their definitions can be found in the Glossary.
Script Text	Indicates text you must enter at a command prompt. Script text also indicates screen text and code examples.
<i>Italics</i>	Indicates variable values you must provide (for example, you may be prompted to supply the name of a <i>file</i> for <i>fileName</i>). Italics also indicate emphasis and the titles of books.
<Return>	Refers to the key on the keyboard labeled with the word Return or the word Enter.
%	Represents the UNIX command-shell prompt for a command that does not require root privileges.
\$	Represents the UNIX command-shell prompt for a command that requires root privileges.
Entering commands	When instructed to "enter" or "issue" a command, type the command and then press <Return>. For example, the instruction "Enter the <i>start</i> command" means type <i>start</i> at a command prompt and then press <Return>.
< >	Indicates a user variable.
[]	Enclose optional items in syntax descriptions.
{ }	Enclose items from which you must make an entry syntax descriptions.
	Separates items in a list of choices enclosed in { } (braces) in code examples. In most cases, spaces are used to separate list items.
...	Ellipsis in syntax descriptions indicate that you can repeat the preceding item one or more times. Ellipsis in examples indicates that information was omitted from the example for the sake of brevity.

Terminology

The following table provides a brief list of terms that are used frequently throughout this guide. For a complete list of terms and their definitions, refer to the [Glossary](#) in this guide.

Term	Description
Community	A community is a set of peers, together with the role that each peer has within that community, and routing information that specifies how packages are routed to each within that community.
Peer	A device (for example, a personal computer or server) that is a member of a community having one of the following roles: community manager, publisher, relay, or target. If a peer belongs to more than one community, it may have a different role in each community.
Community Manager	A community manager is one of the roles that a peer may have. A community manager has the right to create the definition of the community, add or remove peers from a community, and change the roles or routing information of peers within a community.
Publisher	A publisher is a peer that defines the contents of a package and initiates package delivery to a set of targets. After targets open and execute the package, the results are sent back to the publisher in the form of a return receipt. Return receipts are stored and accessed in the publisher's database.
Relay	A relay is a peer that receives a package from either a publisher or another relay and sends the package to targets. Return receipts from the targets are sent to the sending relay, then forwarded to the originating publisher.
Target	A target is a peer that receives a package. When a target receives a package, it opens the package, executes the commands or scripts contained in the package, and sends a return receipt back to the originating publisher.
Package	A set of files and scripts that are delivered to remote computers. All packages are digitally signed by the publisher that originates package delivery. All files and scripts to be delivered are defined by the package specification.
Package Specification	A package specification is an XML file that contains the list of files, scripts, and commands that are used to build and create the package.

Term	Description
Return Receipts	A return receipt is the result of opening a package. Included in the return receipt is the standard output, standard error, and return code of all scripts executed. It also includes the results of applying the file operations. Return receipts are sent by targets to the publishers that originated the package.
Deployment	The process of propagating application files to target systems, then automatically installing the application without human intervention. If a target is unavailable to receive the package, the package is queued until delivery can be made.
Transport	The means by which Everserve systems communicate with each other.

Part 1

Everserve Installation and Configuration

Planning Your Everserve Community

As networks increase in size to accommodate today's information landscape, planning how your network will be used to deliver information becomes increasingly important to ensure an efficient transfer of data. As networks scale, the difficulty in management increases significantly, especially when network objects are geographically distant.

This section discusses the primary tasks and related processes to be considered before configuring your community of Everserve devices. The topics in this section include:

- [Before You Begin](#)
- [Obtaining an Inventory of Devices](#)
- [Designing Your Community](#)

Before You Begin

Before you install Everserve on any of your devices, you will need to consider several factors about your current computing environment. These considerations will affect the success of the installation and operation of your Everserve systems. Getting started with Everserve can be a very exciting experience if you are (properly) prepared.

The first step to success is to plan your Everserve community topology before you install Everserve. This will ensure devices that are intended to have a desired function (or role) in the community are configured correctly and will operate as intended. This section describes how to determine which devices will manage the community objects, which devices will publish packages to community targets, which RMI port to use to transmit messages, how and when to tunnel through firewalls, and other critical details that should be taken into consideration when designing Everserve communities.

Another important detail to consider before installing Everserve is the database you wish to use. Everserve requires each community manager, publisher, and relay in the community have a database installed locally. The section [Setting up the Database](#) in this guide provides information on the databases supported by Everserve, how to configure the database, and other important information critical to a successful installation of Everserve.

Once Everserve has been installed onto a device, you may want to customize the system configuration settings before using Everserve. The section [System Configuration](#) in this guide provides useful information and tips on how to optimize systems for efficient and reliable package delivery.

After installing Everserve on the device that will manage the community, you can create your community of publishers, relays, and targets. Once the community is created and the peers have joined the community, the publisher peer then creates package specifications that define the files, directories, scripts, and commands to send to community peers.

Summary

The following is a summary of the tasks that are key to a successful installation and operation of your Everserve devices. Before you begin installation, consider the following:

- Plan how you want to configure your community. Determine which Everserve devices will act as the community managers, publishers, relays, and targets.
- Obtain hostnames or IP addresses of all systems in the community. This data is needed for the community manager to create peers and to define the community.
- Set-up and configure the database you will use with Everserve.
- Install Everserve onto the device that will act as the community manager, then install Everserve with appropriate role capabilities on all other devices in the community.

Obtaining an Inventory of Devices

Most network operating systems can support an asset management and/or auto-discovery utility that will generate a list of devices currently residing on your network. If you do not have access to an asset management tool that can provide accurate information about the systems in your computing environment, you will need to obtain a snapshot of your network to effectively plan for your Everserve community. With this information, determine:

- Which device will act as the community manager.
- Which device will act as the community publisher.
- Which devices, if any, will be used as relays.
- Which devices will act as targets, receiving and executing packages.
- The logical groupings of devices that would require or expect identical packages.
- How to plan for failover and redundancy to ensure package delivery.
- Which transmission port you will want to use to send packages to the devices in the community.
- Which firewalls, if any, will require configuration to permit Everserve messages to pass between devices separated by the Internet.
- Whether the devices in the planned community meet the minimum system requirements for their intended role.

Designing Your Community

There are several types of community models you can employ when designing your Everserve community. Smaller networks of devices can be designed in less hierarchical topologies where relays may not be needed due to the linear nature of the community design. In cases where large numbers of devices are employed, it becomes increasingly important to have logical and efficient community topologies in place to accommodate network traffic, ensure delivery of all packages, manage wide area load balancing, and minimize the amount of time and effort required to maintain community activity.

Determining Logical Target Groupings

Creating logical groups for your network helps to ensure that only the correct and appropriate packages (containing applications, files, scripts, and commands) are distributed to the intended targets. Logical groups might be groupings of similar operating systems, functional workgroups, or groups of targets requiring access to the same dataset (such as Web page content or applications).

Because Everserve supports heterogeneous computing environments, it is possible to have numerous devices in your network that are running on different operating systems such as a mix of Windows and Solaris devices. To ensure all targets receive packages that are appropriate and understood by their operating systems, you could create two logical groups of targets: one group for Windows targets, and one group for Solaris targets. This type of configuration would ensure that all targets receive only those packages that are appropriate for their operating system.

Using Relays

Using relays is an easy and effective way to group targets together as well as providing a mechanism for mass scalability. A relay sends packages to all targets that are connected to that relay. Using the example described earlier ([Determining Logical Target Groupings](#)), you could configure your community in such a way that a publisher will send different packages to different relays. Each relay would be connected with targets having the same OS. When the publisher sends the (OS specific) package to the relay, the relay then forwards the package to all targets to which it is connected.

[Figure 1](#) illustrates the configuration of the community described above. In this example, the community consists of Windows and Solaris devices that have been logically grouped together using relays.

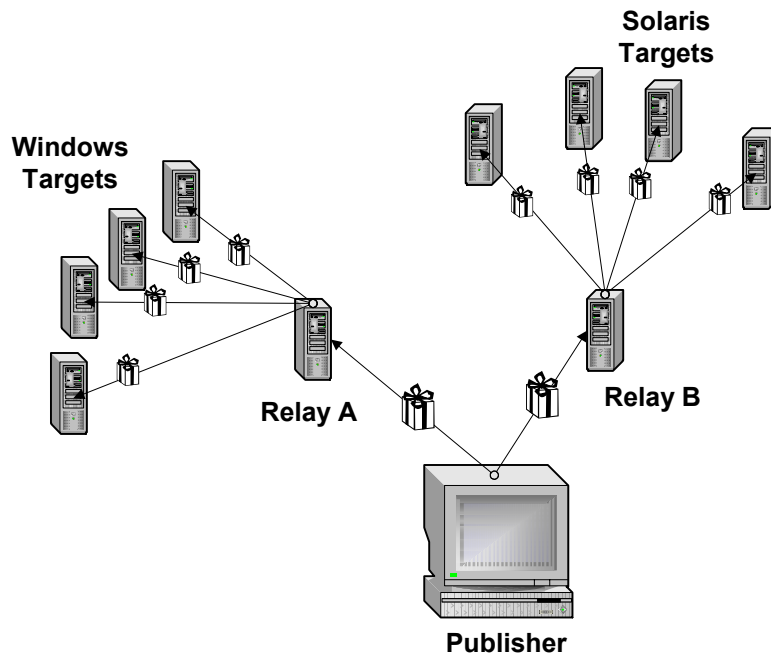


Figure 1 Logical Target Grouping in a Community

Planning for Fail-over and Redundancy

Because network or Internet connections can sometimes fail or be suddenly dropped, you may want to plan a back-up package routing scheme that will ensure all packages are delivered to all targets immediately, rather than having packages held in a queue until the connection is established. You can build redundancy into your Everserve community so that when packages are distributed, they are automatically sent on multiple transport routes to all targets in the community. The first package instance to reach each target is executed on the target, while duplicate packages arriving afterward are ignored.

One example of creating a community with redundancy is illustrated in [Figure 2](#). In this example, the publisher sends the same package to both community relays (relays A and B), then both relays send the package to all targets in the community. The targets will only accept the first arrival of the same package, as determined by the the package's sequence number (each package sent by a publisher has a unique package sequence number).

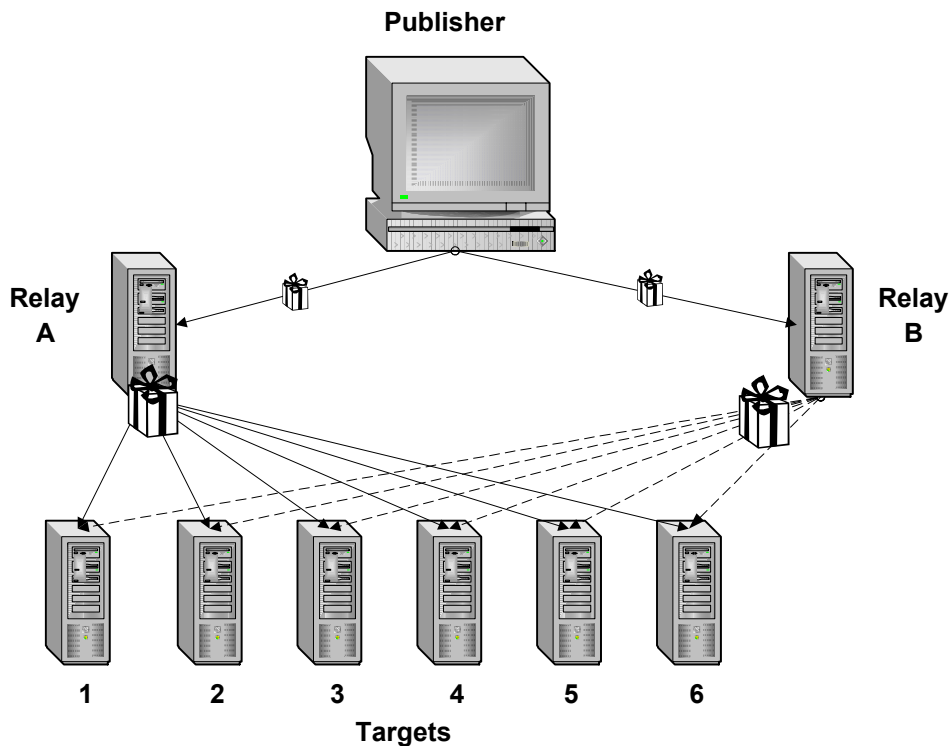


Figure 2 Example, Redundancy Topology using Multiple Relays

Installation

This section provides important information needed to successfully install Everserve. Before installing Everserve, you must plan your community topology, determining details such as which device in the community will act as the community manager, which device will publish packages, which devices will receive and apply packages. These are critical factors that foster the community's configuration and operation. If you have not reviewed the section [Planning Your Everserve Community](#) in this guide, please do so before installing Everserve onto any device.

The topics in this section include:

- [Pre-installation Checklist](#)
- [System Requirements](#)
- [Setting up the Database](#)
- [Installing Everserve on Windows Devices](#)
- [Installing Everserve on Solaris Devices](#)
- [Installing Everserve for Internet Messaging](#)
- [Post Installation Tasks](#)

Note: All devices that are to participate in your Everserve communities must have Everserve installed. A device may participate in a community as a community manager, publisher, relay, or target, but must receive the corresponding installation to function in any of these role-types.

Pre-installation Checklist

To help identify critical information that you will be prompted for during installation, make sure you have the following readily available before you insert the installation CD and begin installing Everserve.

- Which database will be used for each community manager, publisher, and relay in the community?
- Which device will act as the community manager?
- Which device will act as the publisher for the community?
- Which device(s) will act as relays?
- Which device(s) will act as targets?
- Identify the ports will you use for:
 - RMI transmissions (default = 1099)
 - JMS data (default = 1856)
 - JMS administration (default = 1857)
 - Secure HTTPS (default = 8443)
 - Everserve database port (MySQL default = 3306)
- Do the devices in the community meet all recommended system requirements (hardware and software)?
- Will the publisher device use the Everserve Package Editor available on the graphical user interface to create packages? If not does the publisher have an XML editor available to create package specifications? (Although an XML editor is not required, it is recommended for package specification editing.)

System Requirements

The minimum system requirements for each Everserve system depends on the role that system has in the community; where a role can be a community manager, publisher, relay, or target.

Supported Databases

Each community manager, publisher, and relay requires a database. You can choose a database to use at install time, or configure a pre-existing database on your systems with Everserve. Everserve supports MySQL 4.0.2 (recommended), as well as other databases. If choosing to use a database other than MySQL, please contact technical support for assistance to configure the database tables used with Everserve.

Note: You must have the database initialized with a password and running during the Everserve install process.

Supported Platforms

Everserve supports the following platforms:

- Windows NT4 service pack 6a or greater
- Windows 2000 service pack 2
- Windows XP
- Windows 98, 95 (target installs only)
- Solaris 7, Solaris 8, (SPARC)
- Netscape 6.2 and IE 5.5 Web browsers

The following table lists the minimum system requirements for Everserve for each supported operating system and role-type in the community:

Windows Devices	RAM (MB)	Everserve Disk Space (MB)	Database Disk Space (MB)
Community Manager/ Publishers	256 (minimum) 512 (Recommended)	158	60 (minimum) 90 (Recommended)
Relays	256 (minimum) 512 (Recommended)	146	60 (minimum) 90 (Recommended)
Targets	32 (minimum) 64 (Recommended)	38	N/A
Solaris Devices			

Windows Devices	RAM (MB)	Everserve Disk Space (MB)	Database Disk Space (MB)
Community Manager/ Publishers/Relays	256 (minimum) 512 (Recommended)	65	65
Targets	128	65	N/A

Note: All devices regardless of operating system or role-type require SVGA resolution (800 x 600 with 256 color) for GUI installation.

Community Managers, Publishers, and Relays

The internal structure of community managers, publishers, and relays is nearly identical, with each having similar system requirements. Although community managers and publishers have access to Everserve's graphical user interface (Everweb), relays do not. Each of these role-types contain their own JMS Server, transport mechanism, and require a database.

Targets

Targets connect to and use the transport mechanism of a community manager, publisher, or relay to receive packages and to send back return receipts to the originator (community manager or publisher). Targets maintain their persistence store on their own file system and do not require a database. During installation, targets **do not** receive the files necessary to run the Web interface from their device.

Setting up the Database

Each device in the community that is configured as either a community manager, publisher, or relay requires a database. If any of your systems do not already have a database that you can use with Everserve, you can install the MySQL database from the installation CD.

The database must be running prior to Everserve installation. Use the Services Panel on Windows to verify that MySQL is running; on Solaris, look for the MySQL daemon in the process table using the "ps" command.

When installing Everserve, the installer will create a database account with the username "everserve" and the password "everserve." For security purposes, it is recommended that you change the username/password for the database.

Database Security

Regardless of the database you use with Everserve, you should always take necessary precautions to protect your database from unauthorized access, snooping, hacking, and intrusions. Everserve is a deployment tool and does not provide database security—this is the customer's complete and total responsibility. For information and tips on the appropriate measures to take to protect your database,

secure access, assign appropriate user rights and privileges, and change your database username/password, refer to your database manufacturer's documentation.

Database Optimization

If using MySQL as your Everserve database, you can improve the database performance by modifying the default buffer allocations. For information on how to change these settings, visit Synchron's Technical Support Web site at <http://www.synchronnetworks.com/support>.

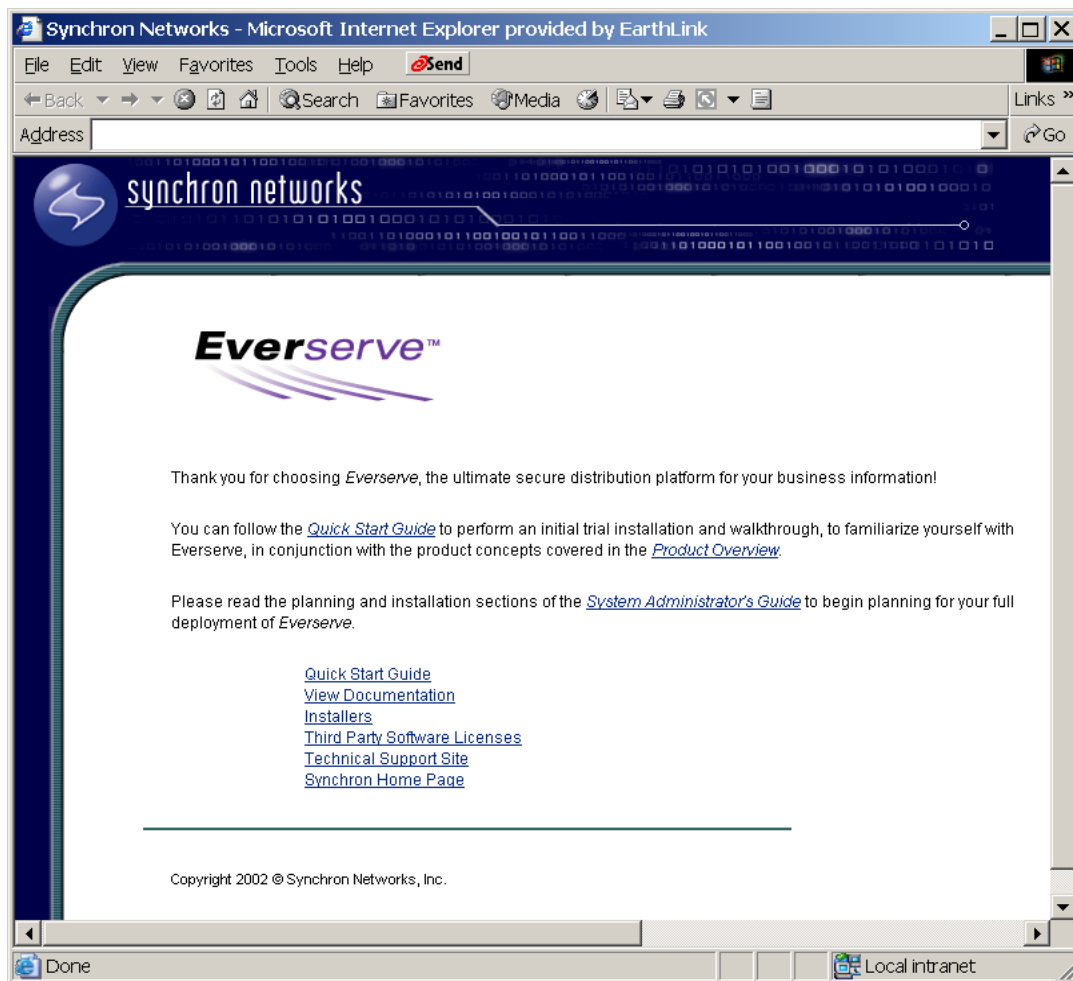
MySQL

MySQL is an optional database supported by Everserve. If you wish to use MySQL for your Everserve database and it is not currently installed on the device, you can choose to install MySQL when installing Everserve onto community manager, publisher, or relay devices in your community.

Installing MySQL on Windows Systems

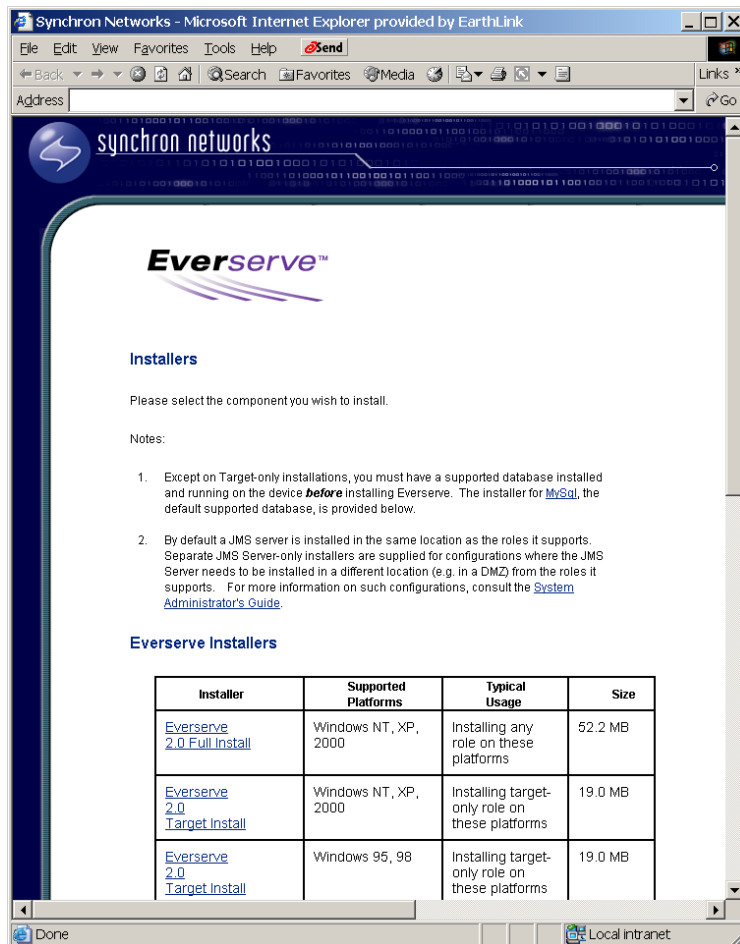
1. Insert the installation CD into the device's CD-ROM drive.

The system displays the Start Here HTML screen:



2. Click **Installers**.

The system displays a component install screen:



3. Scroll down the page and click the **MySQL Installer** link that corresponds to the devices' operating system.
4. Follow the installation and setup instructions provided.

Note: Although Everserve does not require that you secure your database, you may enter a username and password for database security purposes. To do so, follow the instructions in [Step 5](#) as follows:

5. After installing MySQL, go to the MySQL\bin directory and run the winmysqladmin.exe application. Enter a username/password for the MySQL database.

The system will initialize MySQL on the device. A MySQL icon (in the form of a traffic light) will be added to the device's desktop task bar indicating the operational state of MySQL.

Installing MySQL on Solaris Systems

It is recommended that you review the MySQL install documentation before installing MySQL. The MySQL documentation can be found at the following Web site: <http://www.mysql.com>

1. Go to the directory in which to install MySQL and verify there is sufficient disk space for the MySQL installation (/usr/local is used in this example). For example:

```
# cd /usr/local  
# df -k .
```
2. Copy the two files listed below from the /Third Party Software/mysql directory on the installation CD to the directory used for the MySQL install.

Note: The following example uses an 'x' to indicate the Solaris version, either 7 or 8. When issuing the following commands, replace the 'x' (in the x-sparc references) with the appropriate Solaris version running on the device.

```
# cp "/cdrom/everserve2.0/Third Party Software/mysql/mysql-max-4.0.2-  
alpha-sun-solaris2.x-sparc.tar.gz" .  
# cp "cdrom/everserve2.0/Third Party Software/mysql/gtar-sparc.gz" .
```

3. Execute the following commands:

```
# groupadd mysql  
# useradd -g mysql mysql  
# ./gtar-sparc.gz xvfz mysql-max-4.0.2-alpha-sun-solaris2.x-  
sparc.tar.gz  
# ln -s mysql-max-4.0.2-alpha-sun-solaris2.x-sparc mysql  
# cd mysql  
# scripts/mysql_install_db
```

Note: Although Everserve does not require that you secure your database, at this time you may enter a username and password for database security purposes. To do so, follow the instructions displayed on the screen then continue with the MySQL installation as follows:

```
# chown -R root *  
# chown -R mysql data/  
# chgrp -R mysql *  
# chown -R root bin/
```

```
# cp -f support-files/my-medium.cnf /etc/my.cnf
# bin/mysqld_safe --user=mysql &
```

4. Verify the MySQL process is running by issuing the "ps" command. For example:

```
# ps
PID TTY          TIME CMD
27158 pts/1        0:00 ksh
27311 pts/1        0:00 ps
27162 pts/1        0:00 bash
27287 pts/1        0:00 mysqld_s
```

Installing Everserve on Windows Devices

Everserve provides interactive and non-interactive installations for Windows NT4, XP, 2000, and 9x operating systems. The install program allows you to configure devices as either a community manager, publisher, relay, or target. By default, all devices receive target capabilities at install time, which lets you configure any device as a target.

Everserve's interactive installation uses a graphical user interface (GUI) installer, while the non-interactive installation is performed using the silent installers. The GUI installation prompts you for selections and information used to set up Everserve on the device, while silent installations use a properties file that contains the necessary information needed to install Everserve on the device. Properties files must be edited to specify install parameters critical for a complete and successful installation.

*Note: If invoking the installer from a Windows command panel and not the install CD, you must copy **setup.exe** to a temporary directory on the device's hard drive, then run **setup.exe** from this temporary directory.*

Target Only Installations

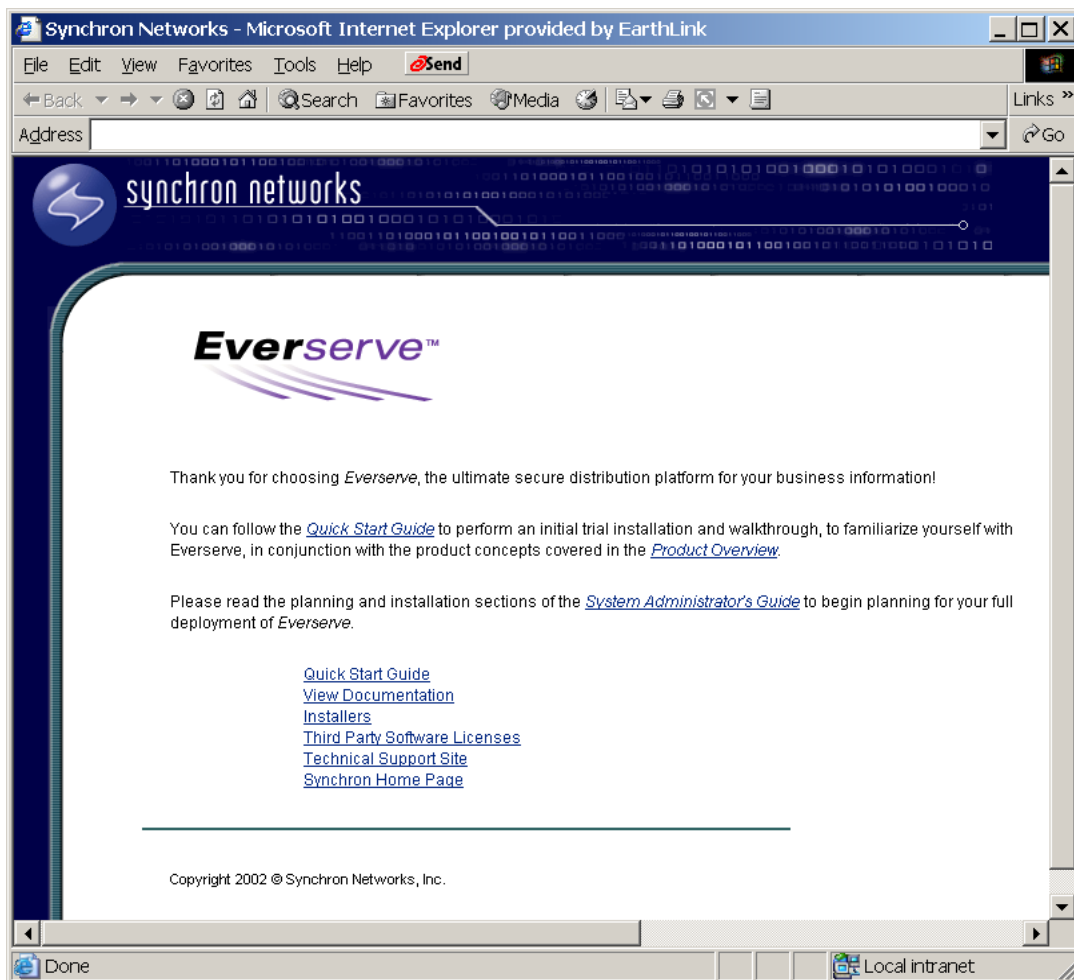
Everserve provides three methods you can use to install a device with target only capabilities: a graphical interface that will install target only capabilities on the device, a graphical custom installer that allows any role type to be installed on the device, and a silent installer that does not require user interaction to install a device with target only capabilities.

The following section describes how to install Everserve onto a device with target only capabilities using the interactive graphical installer. For information and instructions on how to perform a target only installation (or other role capability installations) using the custom installer, see [Community Manager, Publisher, and Relay Installations](#). For information on how to run a non-interactive installation for targets (or other role capability installations), see [Silent Installations](#).

To Perform a Target Only Installation (GUI Install):

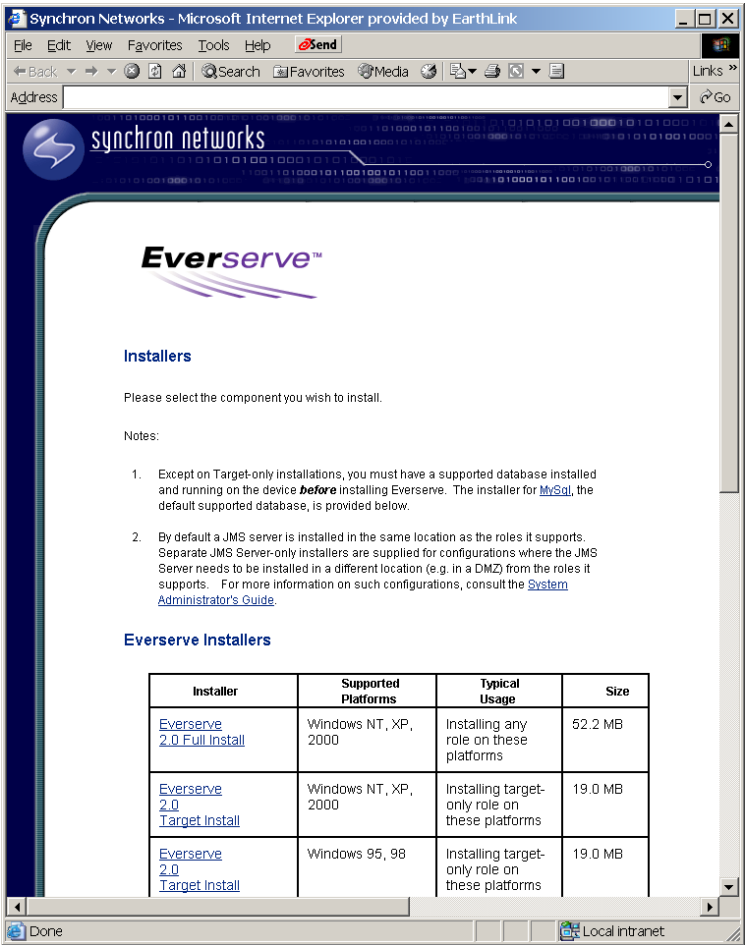
1. Insert the installation CD into the device's CD-ROM drive.

The system displays the Start Here HTML screen:



2. Click **Installers**.

The system displays the following screen:



3. Click the **Windows 2.0 Target Install** link *for the device's operating system* to begin the installation.

The system displays the Everserve License Agreement page:

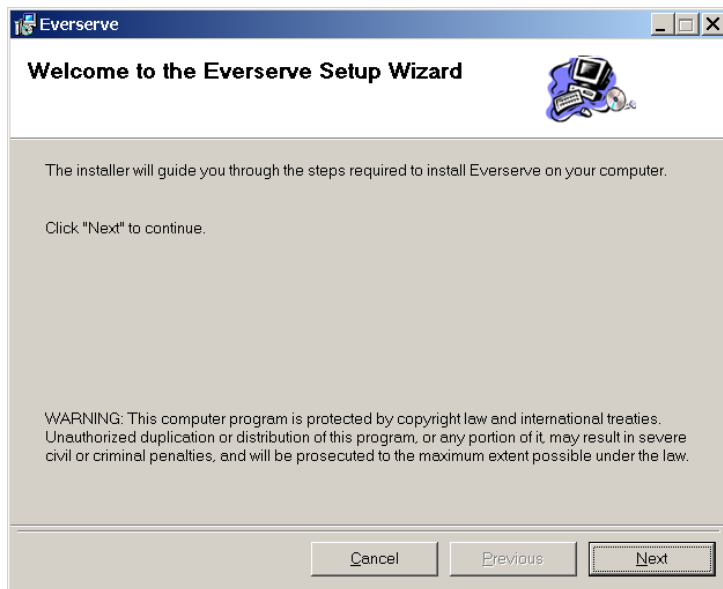


4. After reading and agreeing to the license agreement, click **I Accept the License Agreement** to continue the installation.

The system displays an Explorer window.

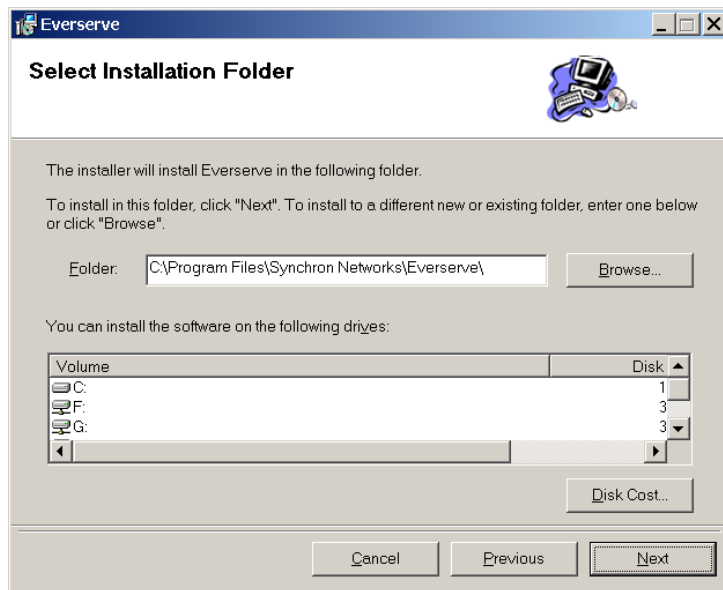
- From the Explorer window, double-click on **Setup**.

The system displays the Setup installation Wizard:



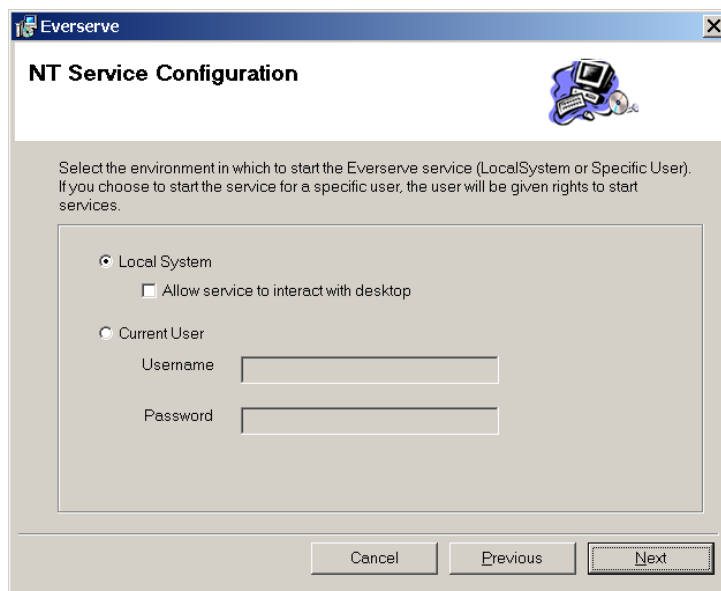
- Click **Next** to continue.

The system displays the following screen:



- Click **Next** to accept the default installation location, or click **Browse** and choose an alternate directory for the installation.

The system displays the following screen:



- If you wish to start Everserve as a service for all users on the device, click **Local System**.

To allow Everserve to interact with the devices' desktop, click the **Allow Service to Interact with Desktop** check box.

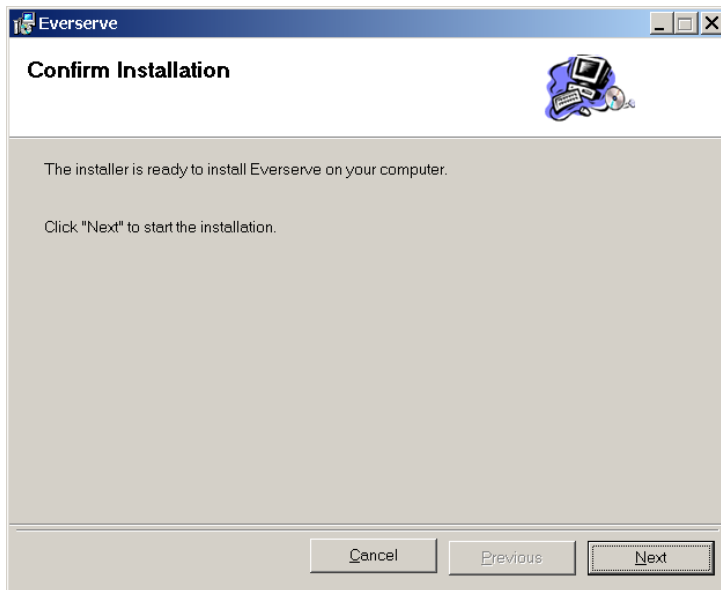
If you wish to enable Everserve only when a specific user is logged into the device, click **Current User** and enter the username and password required.

Note: For Windows 2000 devices, when choosing to start the Everserve service for the Current User, that user must have system level permissions and added to the "log on as a service" group. These permissions are granted from the Administrator Tools> Local Security Policy dialog box.

Giving a user system permissions is not typically advised as that user's account privileges will exceed those of the System Administrator.

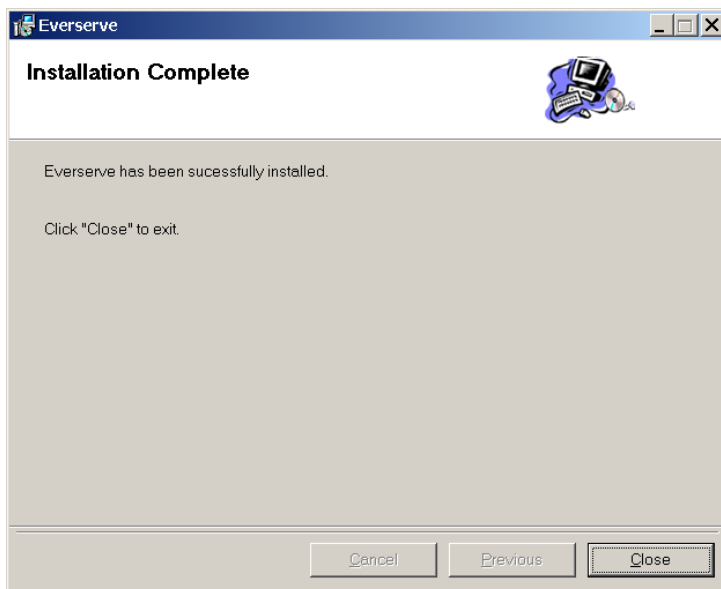
- Click **Next**.

The system displays the following screen:



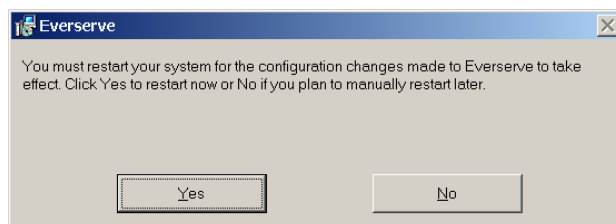
10. Click **Next** to continue the installation.

The system installs all necessary files required for the device to run as a target only in communities. After installation completes, the system displays the following screen:



11. Click **Close**.

The system displays the following screen:



Note: To ensure all configuration settings are invoked properly, you must reboot the device before running Everserve.

12. Click **Yes** to reboot the device at this time, click **No** to reboot manually at a later time.

Community Manager, Publisher, and Relay Installations

The following installation procedure can be used to install any role capability onto Windows devices. For information on how to interactively install Everserve on Solaris devices, see [PKG-ADD Installation for Solaris](#) in this guide.

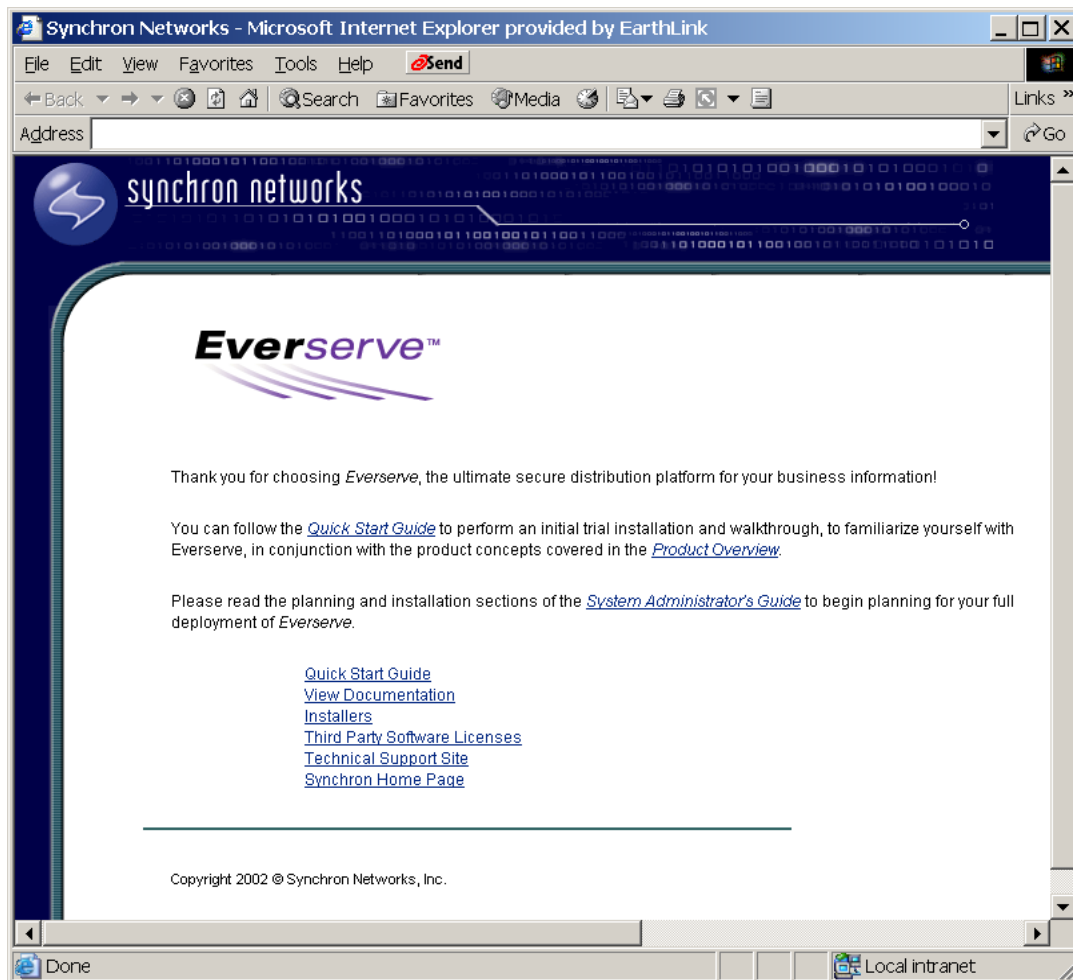
To Run the Windows Custom GUI Installer:

1. Ensure the database you will use with Everserve has been properly installed, initialized, and is running during this installation.

Note: Devices used as a target (only) do not require a database.

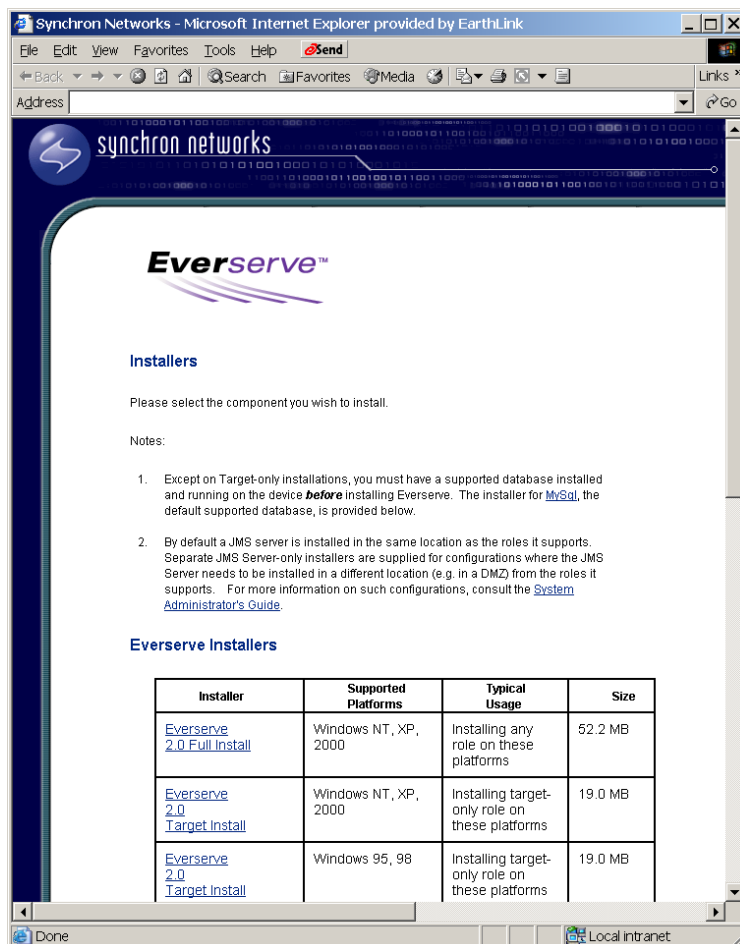
2. Insert the installation CD into the device's CD-ROM drive.

The system displays the Start Here HTML screen:



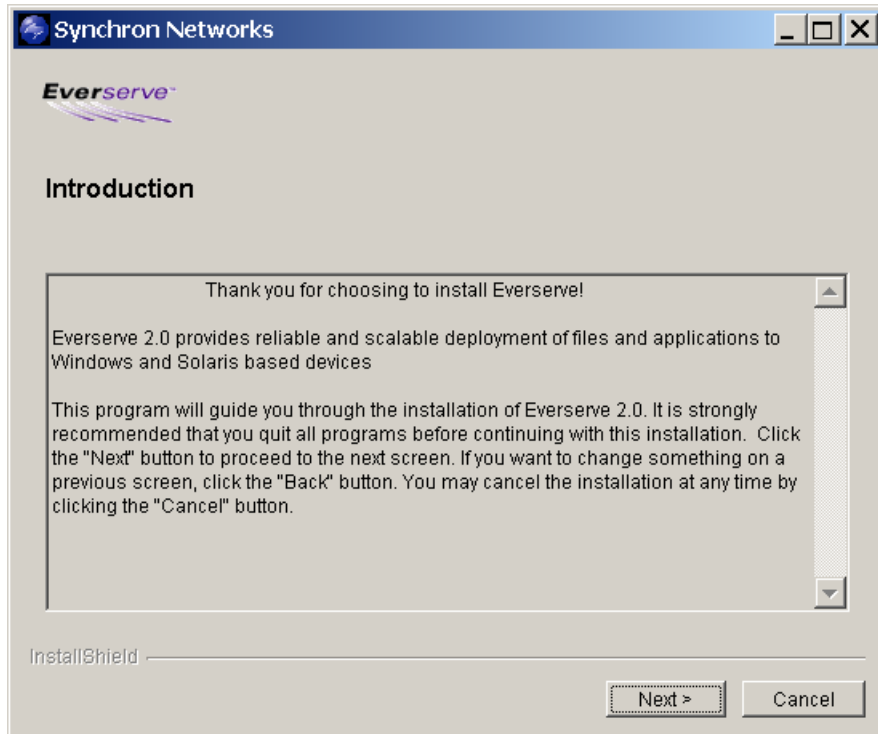
3. Click **Installers**.

The system displays the following screen:



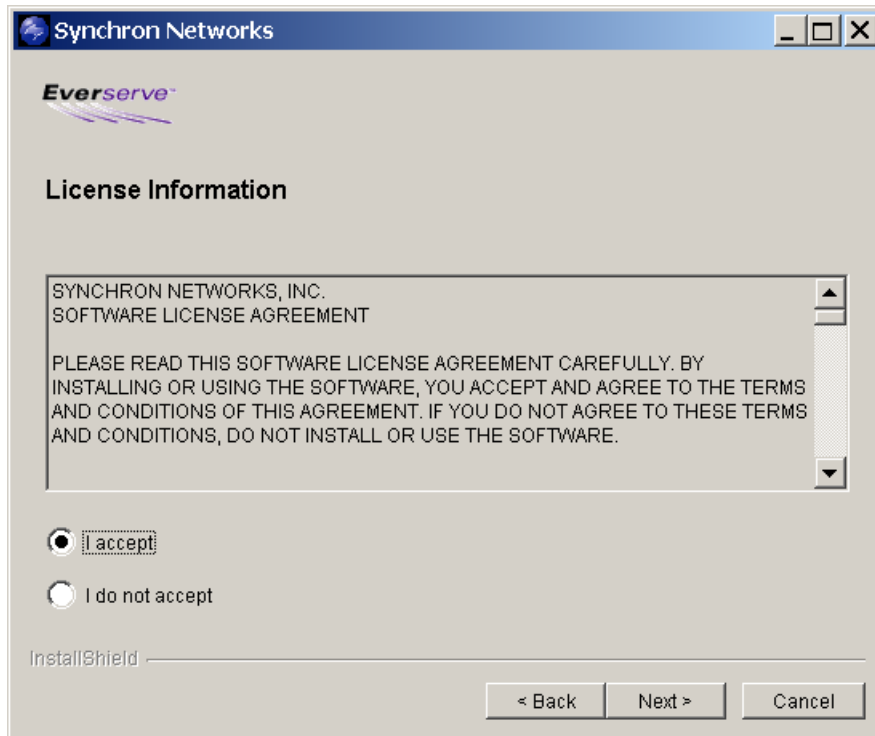
4. Click **Everserve 2.0 Full Install** to begin the installation.

The system displays the following screen:



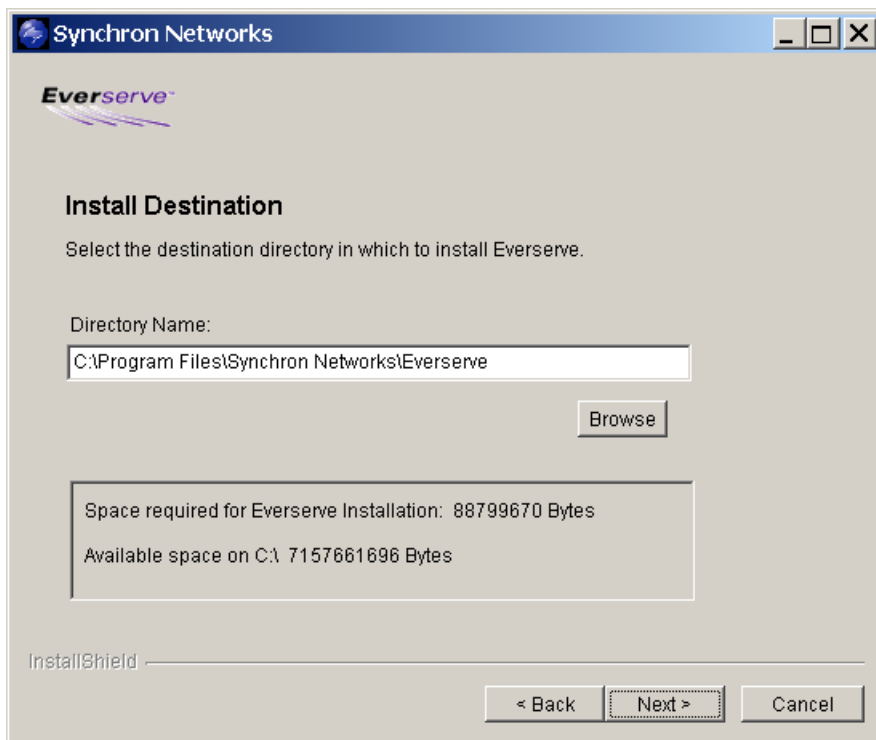
5. Click **Next** to continue.

The system displays the License Agreement screen:



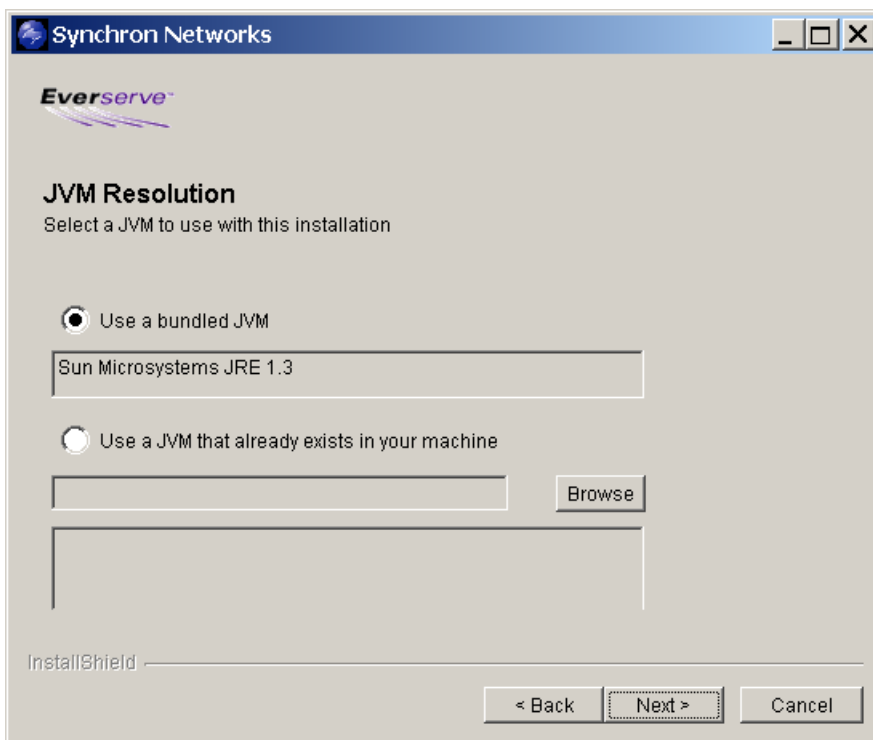
6. After reading and agreeing to the Software License Agreement, select the **I accept** radio button, then click **Next**.

The system displays the Install Destination screen:



7. Specify the directory in which to install Everserve (default installation directory is provided), then click **Next**.

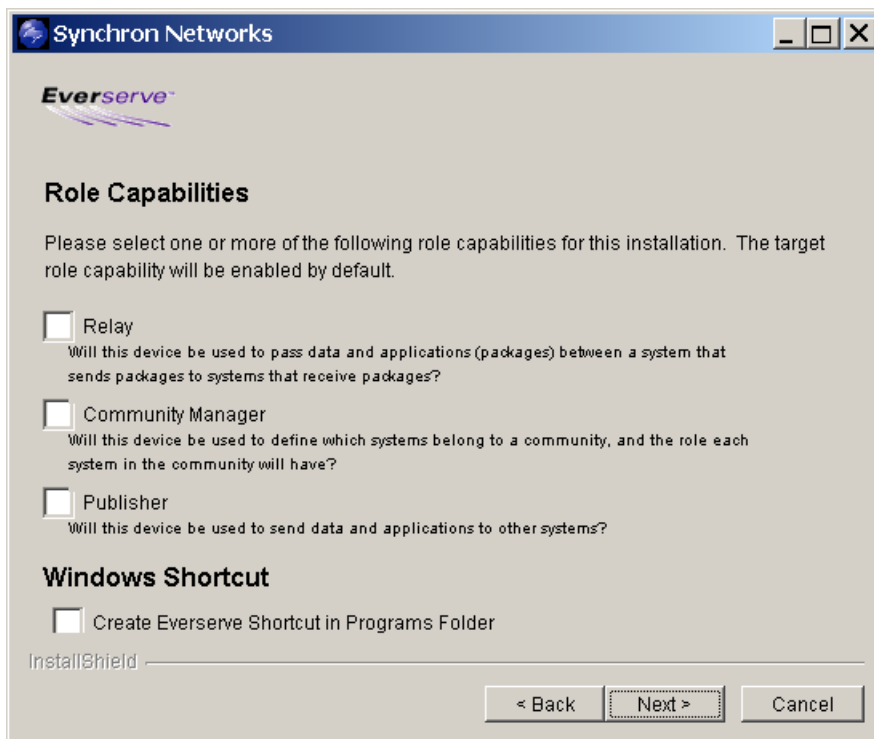
The system displays the JVM Resolution screen from which to select the Java Virtual Machine (JVM) to use with Everserve:



Everserve will display a list of JVMs currently installed on the device. You can choose a JVM from the list to use with Everserve providing the JVM is version 1.3x.

8. Choose the JVM to use with Everserve, then click **Next** to continue.

The system displays the Role Capabilities screen:



9. Choose the role(s) that this device can have in the community. The role-type selected determines which files are installed on the device. Multiple roles may be selected. For example, if you want the community manager and publisher to reside on the same device, select both Community Manager and Publisher from the check list.

The files installed for each role-type in the community are as follows:

Target: Default installation that enables devices to act as a targets. This installation **does not** install sample specifications, Everweb service processes, or documentation. No community manager, publisher, or relay capabilities are installed. Target capabilities are installed on all Everserve devices regardless of role-type selection.

Relay: Installs all files **except** Everweb service processes and sample package specification files. No community manager or publisher capabilities are installed on relay devices.

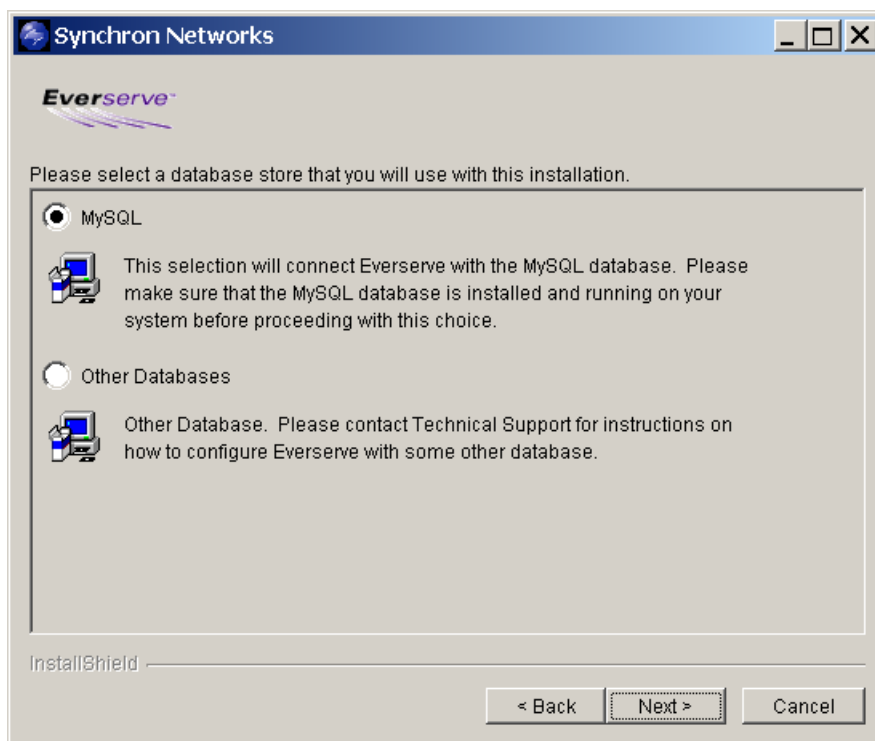
Community Manager: Installs all files and capabilities to manage communities, and includes documentation and Everweb service processes. No publishing capabilities are installed.

Publisher: Installs all files and capabilities needed to deliver packages, and includes sample specifications, documentation, and Everweb service processes. No community manager capabilities are installed.

10. Click **Next** to continue.

11. If you are performing a target only installation, go to [Step 24](#). For all other installations, continue as follows:

The system displays the Database Store selection screen:



12. Select the database to use for Everserve's persistence store, then click **Next**.

Note: If using a database other than MySQL, please contact Synchron Networks Technical Support for assistance with installation and configuration.

When choosing MySQL as the database store, the Everserve Install Wizard will create a MySQL account with the username "everserve" and password "everserve" for the database.

The system displays the Create Everserve Database screen:

Synchron Networks

Everserve™

Database Initialization

This step allows you to create databases used by Everserve. If you have pre-existing Everserve schemas defined in the database store, you may choose to skip this step.

☐ Create Everserve Databases

WARNING: Choosing this option will erase any pre-existing Everserve data, including community settings.

Database Hostname

Database Connection Port

InstallShield

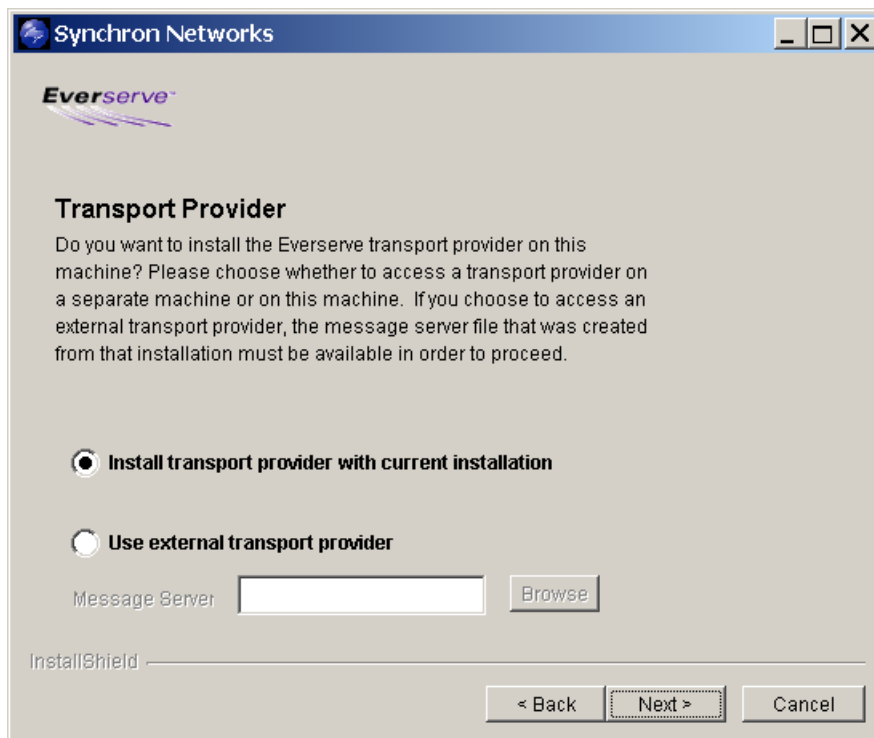
< Back Next > Cancel

13. If this is the first time installing Everserve, the “Create Everserve Databases” check box will be selected. It is used to create the Everserve database tables. If you have an existing Everserve database containing information about your communities, checking this box will erase all existing information contained in the Everserve database.
14. The default home directory is listed for the selected database. If the database is installed in a different location, enter the full path in the Database Home field.
15. The Database Hostname is used to specify name of the device that is hosting the database. Default Database Hostname is the device that is currently receiving the installation.
16. You can specify a Database Connection Port other than the default provided in the Database Connection Port field.

Note: This port must match the MySQL port - it does not change it.

17. Click **Next** to continue.

The system displays the Transport Provider screen:



18. Choose whether to install Everserve's JMS transport provider on this device, or you can choose to use a transport provider located on a remote system.

If choosing **Use External Transport Provider**, click **Browse** and select the `message_server.cfg` file for the remote JMS server you wish to use as the community transport provider. See [Installing Everserve for Internet Messaging](#) for additional information on how to set up a remote JMS server for the community.

The system displays the Transport Configuration screen:

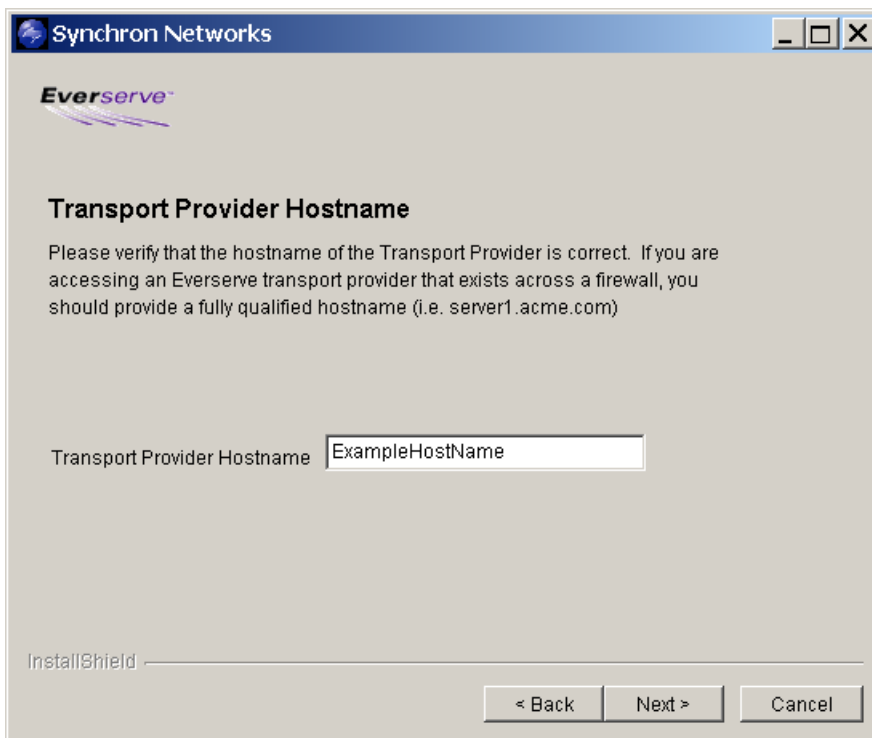
The screenshot shows a window titled "Synchron Networks" with the "Everserve" logo. The main heading is "Transport Provider Configuration". Below it, the text says "Select the RMI Port number for Everserve to use". There is a text box labeled "Everserve RMI Port" with the value "1099". Underneath is the "Fiorano Settings" section, which includes the instruction: "Please make sure that the Fiorano MQServer ports can be exclusively used by this installation." There are two text boxes: "MQServer JNDI Connection Port" with the value "1856" and "MQServer Admin Connection Port" with the value "1857". Below these are two more text boxes: "Admin User Name" with the value "admin" and "Admin Password" with the value "*****". At the bottom left, it says "InstallShield". At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

19. The RMI port is used for internal Everserve administration. By default this port is set to 1099. You may want to change this port setting if you have other applications using this port.
20. The Fiorano Settings are used for JMS traffic and administrative operations. By default these ports are set to 1856 and 1857, respectively. You may want to change these port settings if you have other applications using these ports.

Note: *It is recommended that you install Everserve to run Fiorano with SSL. Failure to do so may compromise the community against hacking, spoofing, or unauthorized access. All communities must be configured with all devices either installed to use SSL or not. With the exception of target only installed devices, Everserve does not allow mixed SSL/non-SSL devices within the same community. To switch devices between SSL and non-SSL, you must reinstall Everserve and choose the appropriate option.*

21. Click **Next** to continue.

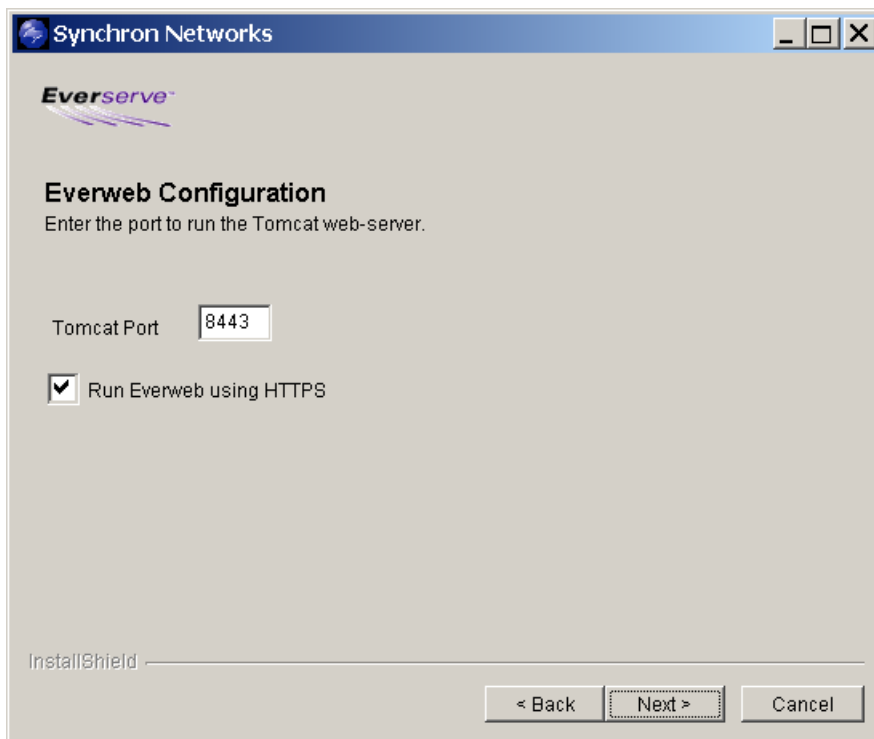
The system displays the Transport Provider Hostname screen:



22. Enter the hostname of the device that will be used as the transport provider. By default, the system fills in the local device's hostname. If you are setting up an Internet Everserve community, provide the fully qualified hostname of the remote JMS server. See [Installing Everserve for Internet Messaging](#) for additional information on how to set up a community for secure Internet transmissions.

If you are installing relay only capabilities, go to [Step 24](#).

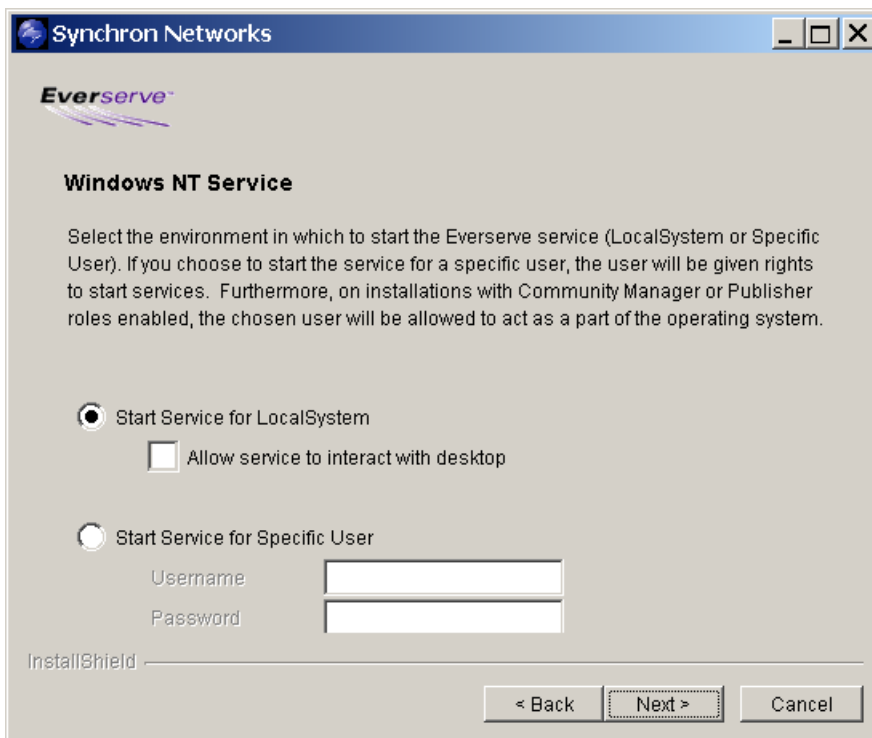
The system displays the Web Server Port assignment screen:



23. Enter the port from which to access the Tomcat Web server, or click **Next** to accept the default port of 8443.

Note: It is strongly recommended that you install Everweb using HTTPS. Failure to do so results in the username and passwords (on the device running Everweb) being passed in clear text.

The system displays the NT Service Authentication screen:



24. If you wish to start Everserve as an NT Service for all users on the device, click **Start Service for Local System**.

To allow Everserve to interact with the devices' desktop, click the **Allow Service to Interact with Desktop** check box.

If you wish to run Everserve with specific user privileges, click **Start Service for Specific User**, and enter the username and password required.

Note: For Windows 2000 devices, when choosing to start the Everserve service for a specific user, that user must have system level permissions, and added to the "log in as a service" group. These permissions are granted from the Administrator Tools> Local Security Policy dialog box.

Giving a user system permissions is not typically advised as that user's account privileges will exceed those of the System Administrator.

25. Click **Next**.

The system displays the Install Summary screen from which to review the installation and setup options:

The screenshot shows a window titled "Synchron Networks" with a sub-header "Everserve". Below this is the "Everserve Installation Summary" section. A message states: "Please make sure that the following information correctly summarizes your preferences." The window contains two main sections: "General" and "Persistent Store".

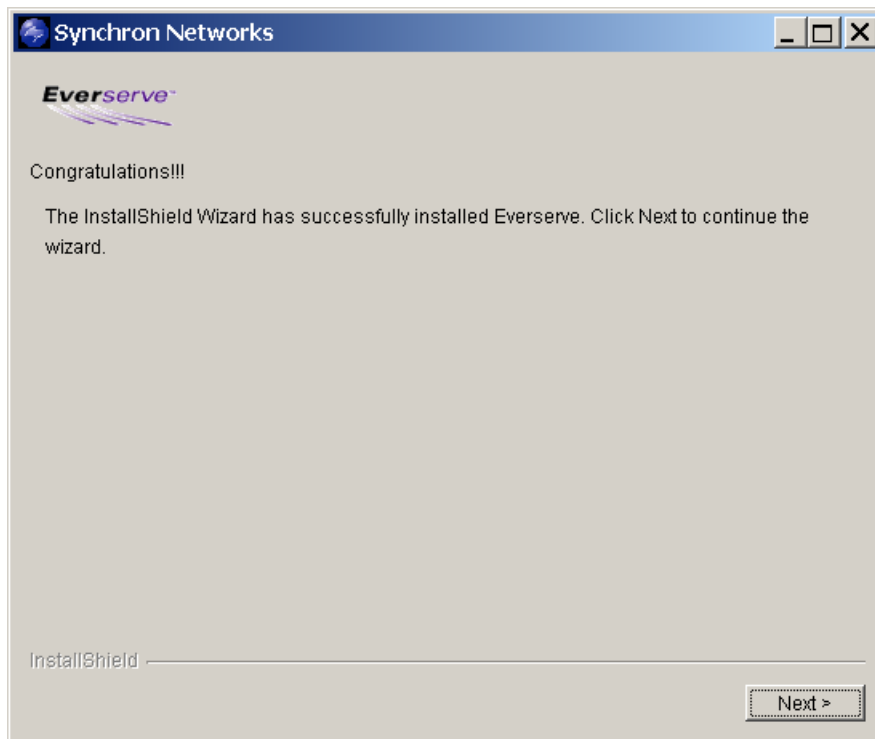
General	
Destination	C:\Program Files\Synchron Networks\Everserve
Roles	target, community manager, publisher
Java Home	C:\Program Files\Synchron Networks\Everserve\jre
Everserve RMI Port	1099

Persistent Store	
DB Manager	MySQL Server
Create Everserve Database?	NO

At the bottom of the window, there is a progress bar labeled "InstallShield" and three buttons: "< Back", "Next >", and "Cancel".

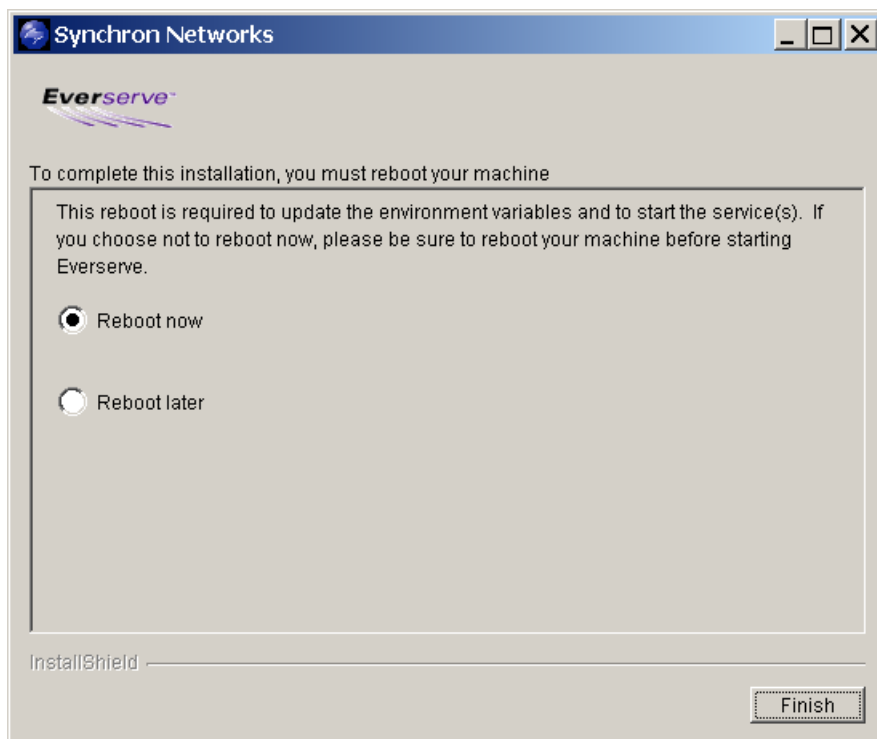
26. If you wish to make changes to the installation and setup, click **Back** to traverse backwards through the installation program and make necessary changes. If the summary of options are correct, click **Next**.

The system installs Everserve and displays the following screen:



27. Click **Next**.

The system displays the following screen:



28. During installation, Everserve updates the Windows registry and other configuration files. You must reboot the device for all changes to take affect. Everserve will automatically reboot the device after installation completes if Reboot Now is selected. If you want to reboot the device at a later time, select the Reboot Later radio button.

Note: If you chose the Reboot Later option, you must reboot your system before running Everserve.

29. Click **Finish**.

Silent Installations

All Everserve silent installations use the same install process regardless of the role being installed on the device. However, what determines the role installed on the device is the properties file (`<peerRole.Properties>`) used during the installation. The Everserve install CD provides a silent installation properties file for each role that can be installed on a device (community manager, publisher, relay, and target).

As a minimum, all silent installations will include the basic files needed to run Everserve, receive packages, and send return receipts back to the device that originated the package regardless of the role-type installation.

Regardless of the role you are installing, you must copy the installer **setup** executable program and the installer.properties (`<peerRole.properties>`) files to a temporary directory on your hard disk, then run the `<peerRole.properties>` file from this directory. These files are located at:

Installers\Windows\WinNTCustomInstall\setup.exe and **`<peerRole.properties>`**

The following section describes how to run silent installations on Windows devices used as targets, relays, publishers, and community managers.

Note: Before attempting a silent installation, verify that all settings in the `<peerRole.properties>` file are correct, and if not, modify the file as needed.

To Perform a Silent Installation:

1. Copy the files **setup** and **`<peerRole.Properties>`** from the installation CD (located in `\Installers\Windows`) to a temporary directory on the device's hard disk.

The `<peerRole.properties>` will be any or all of the following files:

- cmPub.properties (for both community manager and publisher capabilities)
- communityManager.properties
- publisher.properties
- relay.properties
- target.properties

2. From a command window, change to the temporary directory containing the **setup** and **`<peerRole.properties>`** files and enter the following command:

```
setup.exe -options <peerRole.properties>
```

Where Options =

Install Destination:

```
-P absoluteInstallLocation="<Install Location>"
```

Java Home:

```
-W jvmResolution.chosenJavaHome="<Java Home>"
```

3. Reboot the device after the installation process completes before running Everserve.

Sample peerRole.properties File

```
#####
# Property file for community manager with publisher installation
# Creation Date: 9/20/2002
#Revision 1.0
#####
# Specify silent mode
#####
-silent
#####
# Everserve shortcut in the Start Menu
#####
# -W destination.installShortcut="False"
#####
# Role types.
# For target-only install, comment out all roles below.
#####
# relay:
#-W roleCapability.relay="True"
# community manager:
-W roleCapability.communityManager="True"
# publisher:
-W roleCapability.publisher="True"
#####
# Persistent storage type.
# Not applicable for target-only installs.
#####
-W customFeatures.dbStore="dbMySQL"
#####
# Create Everserve databases
# Not applicable for target-only installs.
#####
```

```
-W createDatabase.createDatabases="True"
#####
# Reboot machine after installation is complete
#####
-W rebootPanel.reboot="False"
```

Installing Everserve on Solaris Devices

Everserve provides interactive and non-interactive installations for Solaris 7 and 8 operating systems. The install program allows you to configure devices as either a community manager, publisher, relay, or target. By default, all devices receive target capabilities at install time, which lets you configure any device as a target.

Everserve's interactive installation (PKG installer) prompts you for selections and information used to set up Everserve on the device, while silent installations use a response file that contains the necessary information needed to install Everserve on the device. Response files must be edited to specify install parameters critical for a complete and successful installation.

The following sections describe how to perform interactive and silent Solaris installations.

PKG-ADD Installation for Solaris

The PKG installer is used to install Everserve onto Solaris devices. Device role capabilities, database configuration, and applicable TCP port designations are all specified during the installation. For information on how to perform silent, non-interactive installations, see [Silent Installations](#).

Note: Please read the license agreement on the installation CD before proceeding with this procedure.

Summary

This information in this section summarizes the PKG installation procedure. Detailed information for each of these tasks is provided in the sections that follow.

1. Uninstall any prior versions of Everserve before installing this release.
2. Install a database for use with Everserve if one does not currently exist. A database is required if the device will be used as a community manager, publisher, or relay. Targets do not require a database. Instructions for installing a MySQL database are included in this guide; refer to the section [Installing MySQL on Solaris Systems](#) for instructions on installing MySQL.
3. Install Everserve.

Uninstall Prior Versions of Everserve

1. Login as root.
2. Determine if a previous version of Everserve is installed on the device. The easiest way to determine the state of the system is to issue the "everserve version" command and observe the response of the system.

```
# everserve version
```


If Everserve is installed on the device, the system will display the product version. In this case, proceed to [Step 3](#).

If Everserve has not been installed on the device the message "everserve: command not found" (or similar) will be returned. In this case, check the directory in which you wish to install Everserve and verify the /Synchron directory does not exist. If a /Synchron directory is present you must remove it before continuing with this procedure. Once it is determined that Everserve is not installed on the system, skip the following procedure and go directly to [Installing MySQL on Solaris Systems](#) in this user guide.

3. Issue the Stop command to shutdown Everserve:

```
# everserve stop
```

4. Determine the type of installation of the existing Everserve version:

```
# ls /usr/local/synchron | grep _uninst
```

If the _uninst directory is present, Everserve was previously installed using the setup.bin installer.

If the _uninst directory does not exist, Everserve was installed using the PKG-ADD program.

5. Using the results from [Step 4](#), perform one of the following steps:

- If the directory _uninst exists, go to the install directory and uninstall Everserve as follows:

```
# cd /usr/local/synchron/_uninst
```

```
# ./uninstall.bin
```

- If the /_uninst directory does not exist, enter the following command to uninstall Everserve:

```
# pkgrm Everserve
```

6. After Everserve has been uninstalled, delete the /synchron directory.

Note: If you wish to save any previous files in the /synchron directory, save them to a separate location.

```
# cd /usr/local
```

```
# rm -rf synchron
```

Installing MySQL

Before installing Everserve, you must set up the database. Follow the instructions in the section [Installing MySQL on Solaris Systems](#) earlier in this guide for database set-up information and instructions.

Installing Everserve

1. Go to a temporary directory in which to begin the Everserve installation. For example:

```
# cd /usr/tmp
```

2. Copy the Everserve-2.0-3.22.pkg.tar file from the CD to the temporary directory.

```
# cp /cdrom/everserve2.0/Installers/Solaris/Everserve-2.0-3.22.pkg.tar .
```

3. Untar the Everserve file:

```
# tar -xvf Everserve-2.0-3.22.pkg.tar
```

4. Install Everserve from the temporary directory:

```
# pkgadd -d .
```

The system will display a list of all packages available for installation.

5. Enter the PKG instance you wish to install:

```
# Everserve
```

The system will display the Synchron Networks Application Platform license agreement and applicable copyrights.

6. If you agree to the terms and conditions in the license agreement, enter 'y' when prompted:

```
Do you accept the terms of this license agreement? [n] [y,n,?,q] y
```

7. By default Everserve will be installed in the /opt directory. If you wish to install Everserve into a different directory than the default, enter the directory path:

```
Please choose an install directory [/opt]
```

8. If you want to define a user account that has privileges to access Everserve, enter that account name when prompted (default =root).

```
Type in the user that you want to run and own Everserve processes [root]
```

9. To have Everserve start automatically after a device reboot, accept the default provided, or choose 'n' so that Everserve does not automatically start after a reboot. (in this case, you must manually start Everserve each time the device is restarted).

```
Would you like everserve to start automatically at boot time? [y]
[y,n,?,q]
```

10. Everserve requires a JRE 1.3 or newer. A compatible JRE is provided with the installer:

```
Would you like to install the JRE bundled with Everserve? [y] [y,n,?,q]
```

11. If you would like this device to pass messages between the publisher and designated peers, choose 'y' at the following prompt:

```
Would you like to have Relay capability for this installation of
Everserve? [n] [y,n,?,q]
```

12. If this device will be used to create communities, peers, and assign peer roles within a community, choose 'y' at the following prompt:

```
Would you like to have CM capability for this installation of Everserve?
[n] [y,n,?,q]
```

13. If this device will be used to create and deliver packages to peers, choose 'y' at the following prompt:

```
Would you like to have Publisher capability for this installation of
Everserve? [n] [y,n,?,q]
```

14. If you wish to configure your Everserve community to share a remote JMS server (used for secure Internet messaging), choose 'y' at the following prompt:

```
Do you want to use a transport provider on a different machine [n]
[y,n,?,q] n
```

15. Enter the hostname of the machine that will be used for JMS (default is the local hostname)

```
Please specify the fully qualified host machine name where the transport
provider is located [my.example.com]
```

16. If you would like to run SSL for secure transmissions, choose 'y' at the following prompt:

```
Please choose whether or not to run Fiorano in a secure mode [y]
[y,n,?,q] y
```

17. Specify the JMS port to use for Everserve transmissions:

Please choose a JNDI port for Fiorano MQServer connection [1856]

18. Specify the JMS port to use for administration transmissions:

Please choose an Admin port for Fiorano MQServer connection [1857]

19. Specify a JMS administration username:

Please enter a logon name to use for Fiorano admin connections [admin]

20. Specify the password used with the JMS administration username:

Please enter a logon password to use for Fiorano admin connections
[passwd]

21. Devices configured as a community manager, publisher, or relay require a database. If you wish to use MySQL as the database for this device, choose 'y' at the following prompt. If you wish to use a database other than MySQL, please contact Synchron's Technical Support for assistance with configuring the database with Everserve.

Would you like to use MySQL for persistent storage? [y] [y,n,?,q]

22. Enter the full directory path where the MySQL database is located:

Where is MySQL located? [/usr/local/mysql]

23. Enter the full directory path where the MySQL data directory is located:

Where is the data directory? [/usr/local/mysql/data]

24. If this is a first time installation, choose 'y' to the following prompt, If you already have an Everserve database and wish to keep all existing data, choose 'n' to the following prompt:

Would you like to create the database? [y] [y,n,?,q]

25. Enter the hostname where the database is located:

Specify the host where the database resides? [localhost]

26. Enter the port to use for the database connection:

Specify the port to use for database connection? [3306]

27. By default, Everserve uses port 1099 as the default TCP port for RMI. If you wish to use a different port for this purpose, enter the port number you wish to use.

Please choose a TCP port for RMI [1099]

28. Specify if you would like to use a secure HTTP connection with Everweb:

Would you like to use a secure HTTPS connection when operating Everweb?
[y]

[y,n,?,q]

29. By default, Everserve uses port 8443 as the default TCP port for its Web client, Everweb. If you wish to use a different port for this purpose, enter the port number you wish to use.

Please choose a TCP port for Everweb [8443]

The system displays a summary of all settings selected for this installation. Ensure that all settings are correct before proceeding with the installation:

```

INSTALL_DIR      /opt
JavaHome         /opt/synchron/jre
Install JRE?     yes
Role             target, relay, community manager, publisher
Persistent Store MySQL
Initialize DB    yes
RMI Port         1099
ExternalTransport no
Transport Host   my.example.com
SSL Seed File    /tmp/message_server.cfg
Use Fiorano SSL  yes
Admin Username   admin
Admin Password   passwd
Fiorano JNDI     1856
Fiorano Admin    1857
Use HTTPS?      yes
Everweb Port     8443
Process Owner    root
AutoStart        yes

```

```
MySQL Dir      /usr/local/mysql
MySQL Data     /usr/local/mysql/data
Database Host  localhost
Database Port  3306
```

30. If any of the installation settings are incorrect, choose 'n' at the following prompt. If all settings are correct, choose 'y' at the following prompt.

```
Are these parameters correct? [y]  [y,n,?,q]
```

31. To continue the installation, enter 'y' at the following prompt:

```
Do you want to continue with the installation of <Everserve> [y,n,?] y
```

32. The system installs Everserve using the specified settings. When the installation completes, the system will display the following message:

```
Installation of <Everserve> was successful.
```

The system will install the following subdirectories in /opt/synchron:

```
/bin
/lib
/server
/external
/jre
/db
```

Note: The subdirectory /external will not be installed on devices receiving relay or target only capability installs.

33. Start Everserve:

```
# everserve start
```

34. Start an Everserve session:

```
# everserve
```

The system displays the following message, then displays the Everserve prompt:

```
Checking Everserve server...
```

```
Everserve 2.0-3.22 Copyright (c) 2001-2002, Synchron Networks, All  
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

```
Everserve>
```

Silent Installations

All Everserve silent installations use the same install process regardless of the role being installed on the device. However, what determines the role installed on the device is the response file (**<peerRole.response>**) used during the installation. The Everserve PKG installer includes response files used for performing silent installations. The response files are used to specify the role-type being installed on the device, where role-type can be community manager, publisher, relay, or target. Before attempting a silent installation, verify all settings in the **<peerRole.response>** file contains the desired settings, and if not, modify the file as needed.

As a minimum, all silent installations will include the basic files needed to run Everserve, receive packages, and send return receipts back to the device that originated the package regardless of the role-type installation.

The following section describes how to run silent installations on Solaris devices used as targets, relays, publishers, and community managers.

Note: Before attempting a silent installation, verify that all settings in the <peerRole.response> file are correct, and if not, modify the file as needed.

To Run the Solaris Silent Installer:

1. Log into the device as root.
2. Uninstall any prior version of Everserve that may be present on the device.
3. Create a temporary directory in which to copy the PKG installer from the install CD. For example:

```
# mkdir /usr/tmp/everserve_pkg
```
4. Go to the temporary directory created in [Step 3](#):

```
# cd /usr/tmp/everserve_pkg
```
5. Insert the Everserve Installation CD into the device's CD-ROM drive and copy the PKG installer to the temporary directory created in [Step 3](#):

```
# cp /cdrom/everserve2.0/Installers/Solaris/Everserve-2.0-3.22.pkg.tar
```
6. Enter the following command to tar the Everserve installer:

```
# tar xvf Everserve-2.0-3.22.pkg.tar
```


7. Ensure all settings specified in the `<peerRole.response>` file used for this installation are correct. For example, **the default installation directory is /opt**. If you wish to install Everserve into a directory other than /opt, you must edit the `<peerRole.response>` file and specify the directory in which to install Everserve.

The following example shows the default settings for the `cmPub.response` file. This response file will install Everserve in the /opt directory (shown as `BASEDIR=/opt`), and install target, community manager, and publisher role capabilities for the device (shown as `Roles=target, community manager, publisher`).

Sample `cmPub.response` file:

```
DBASE=
MYSQL_HOME=/usr/local/mysql
MYSQL_DATA=/usr/local/mysql/data
DB_HOST=localhost
DB_PORT=3306
BASEDIR=/opt
JavaHome=/opt/synchron/jre
JavaInstall=yes
Roles=target, community manager, publisher
PersistentStore=MySQL
InitializeDB=yes
UseHttps=yes
TomcatPort=8443
SecureFiorano=yes
JNDIPort=1856
AdminPort=1857
FMP_HOME=/opt/synchron/FioranoMQ5.3
RMIPort=1099
AutoStart=yes
ProcessOwner=root
```

8. Once all settings in the **<peerRole.response>** file have been verified for this installation, start the silent install process as follows:

```
# pkgadd -a silent.admin -r $PWD/<peerRole.response> -d $PWD Everserve
```

The installer will install Everserve using all settings defined in the **<peerRole.repsonse>** file and create the following subdirectories from the /opt/synchron (or specified install) directory:

\bin	\lib	\server
\external	\jre	\db

Note: The subdirectory /external will not be installed on devices receiving relay or target capability only installs. The /external subdirectory is only installed for community manager or publisher installations.

9. Start Everserve:

```
# everserve start
```

10. Start an Everserve session:

```
# everserve
```

The system displays the following message, then displays the Everserve prompt:

```
Checking Everserve server...
```

```
Everserve 2.0-3.22 Copyright (c) 2001-2002, Synchron Networks, All  
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

```
Everserve>
```

Installing Everserve for Internet Messaging

Configuring Everserve for secure Internet messaging requires an appropriate community manager/publisher device to be located in the DMZ that will be designated for JMS messaging and transmissions in the community. This JMS server is configured with JMS messaging software only — it does not have Everserve capabilities installed. This JMS server is a passive server, meaning it does not make any outgoing connections.

All devices located on both sides of the firewall communicate with each other through the dedicated JMS server. The community managers and publishers are located behind the corporate firewall and communicate to externally located devices using this community JMS server. Any device that is a member of an Internet Everserve community communicate with the community managers, publishers, and relays using this community JMS server. [Figure 3](#) illustrates a simple Internet configuration for an Everserve community.

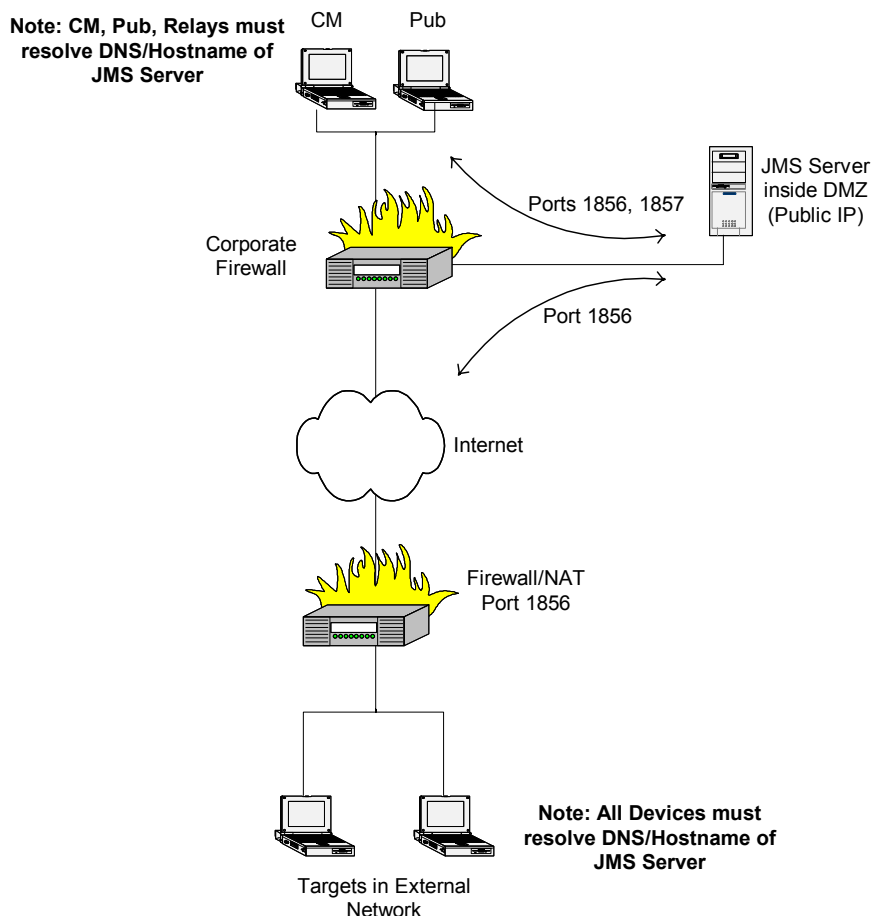


Figure 3 Internet Transmission Configuration

Configuration Summary

The following summarizes the process used to configure Everserve for secure Internet messaging:

1. Configure the firewall/DMZ so that target systems outside the firewall can attach to the first of the two JMS server ports (1856 by default). Community managers, publishers and relays that use this JMS server will need to attach to both JMS server ports (1856 and 1857 by default).
2. Install only JMS server on the device that will be used for all Everserve transmissions. During this installation, the publicly routable IP address and publicly routable hostname for this JMS server is identified and included in the `message_server.cfg` file.
3. Install Everserve on the community manager, publisher, and relays using `message_server.cfg` to specify JMS transmission address and ports used by the dedicated JMS server.
4. Install Everserve on all targets using the community seed generated by the community manager.
For assistance in setting up your Everserve network for secure Internet messaging, please contact Synchron Networks Professional Services group.

JMS Server Only Installations

To set up Everserve to run securely over the Internet, you must have a device that is installed with JMS only in the DMZ. This device will be used as the community's JMS server, passing messages between the Intranet located behind a firewall to devices located globally. All devices located behind the firewall pass messages through the firewall to the remote JMS server, while devices located remotely (accessed over the Internet) pass messages to the remote JMS server, which forwards the message to devices located behind the firewall.

Note: The device used as the JMS server is installed with JMS capabilities only — it does not receive an installation of Everserve software.

The following sections describe how to install JMS only on Windows and Solaris devices.

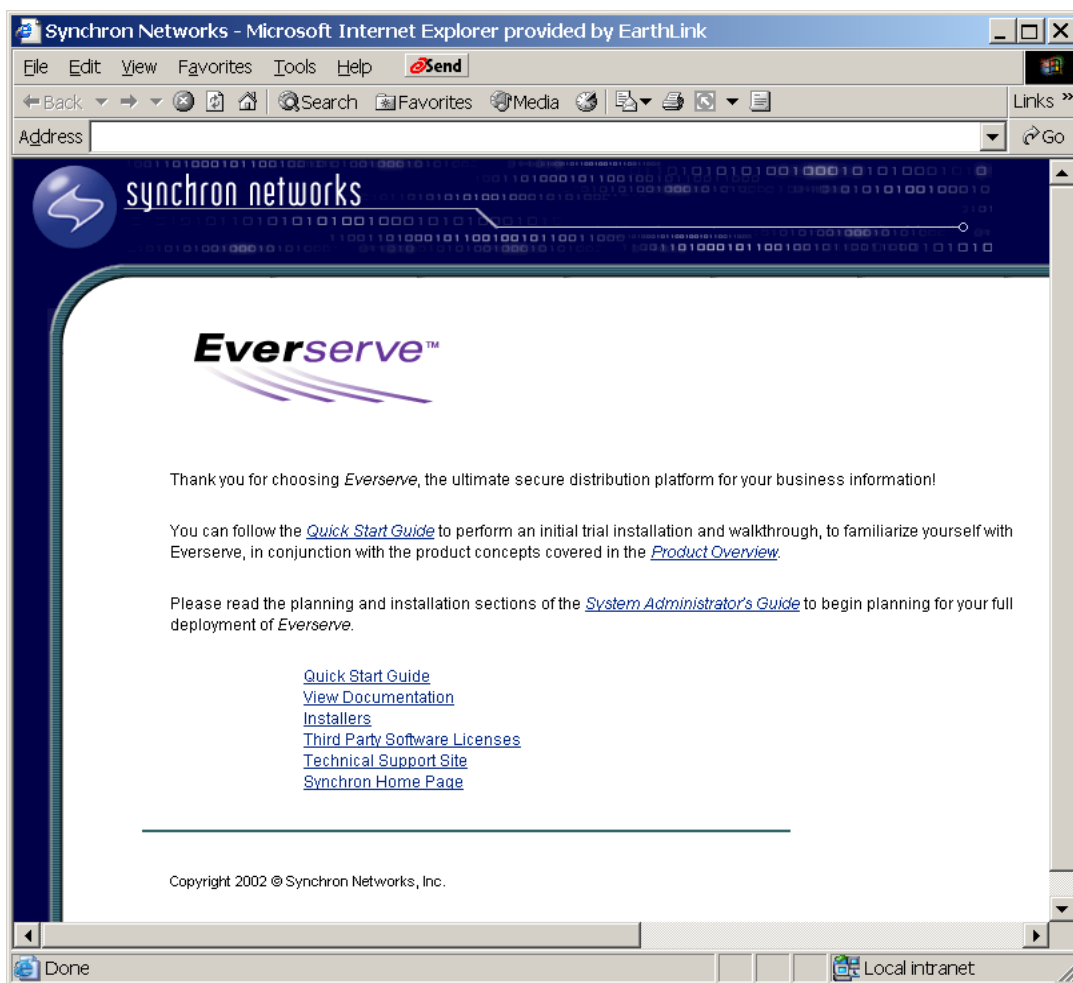
Windows JMS Only Installations

JMS only installations can be performed on Windows NT, XP, and 2000 devices only. Due to resource limitations, Windows 95 and 98 devices can only be configured as targets and should not be installed with JMS only software.

To Perform a JMS Only Installation:

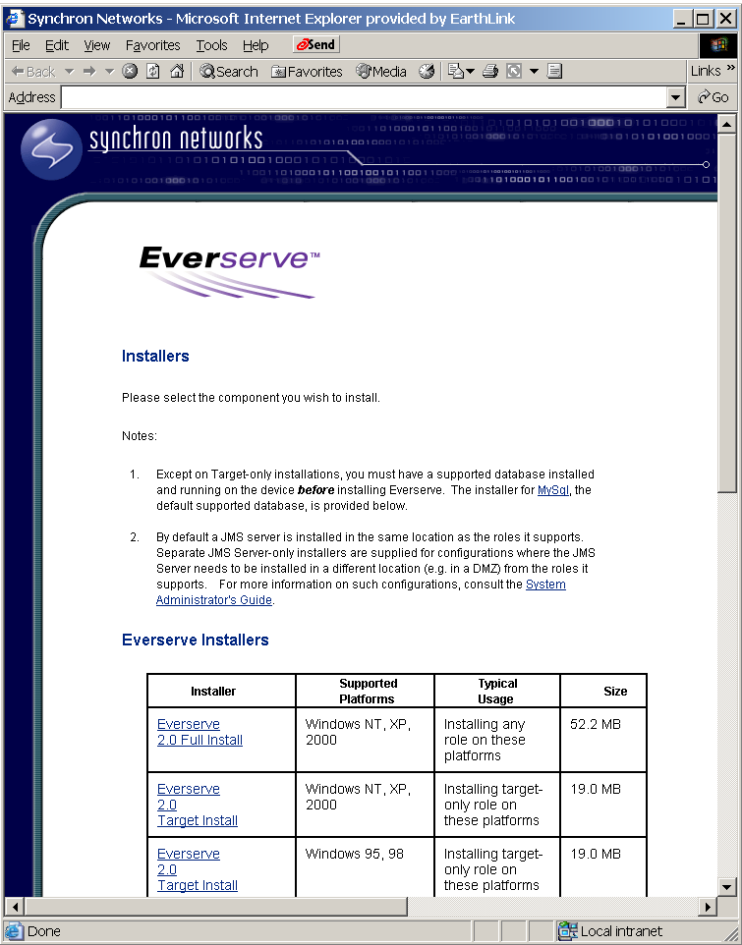
1. Insert the installation CD into the device's CD-ROM drive.

The system displays the Start Here HTML screen:



2. Click **Installers**.

The system displays the following screen:



3. Scroll down the page and click the **JmsServer** link for the device's operating system to begin the installation.
4. The system displays an Explorer window.

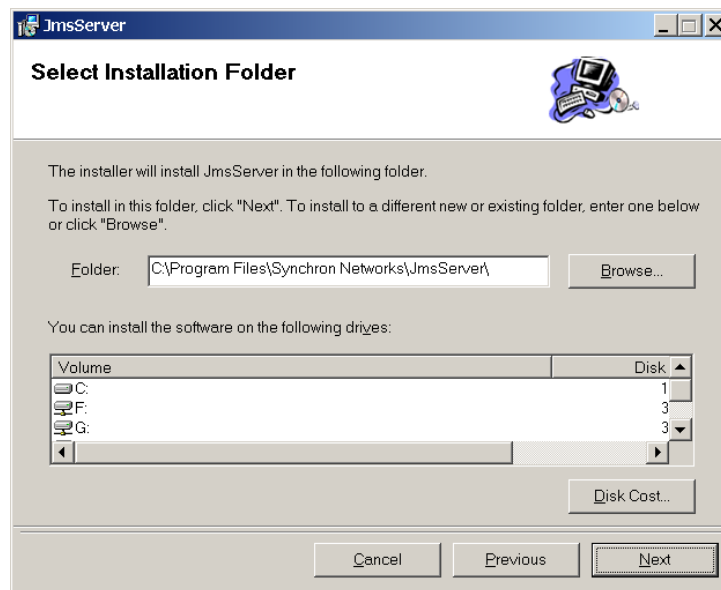
- From the Explorer window, double-click on **Setup**.

The system displays the JmsServer Setup installation Wizard:



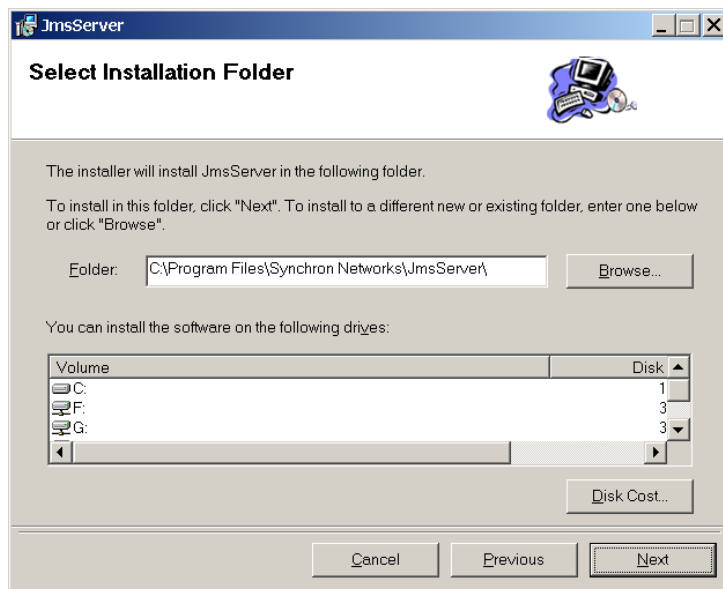
- Click **Next** to continue.

The system displays the following screen:



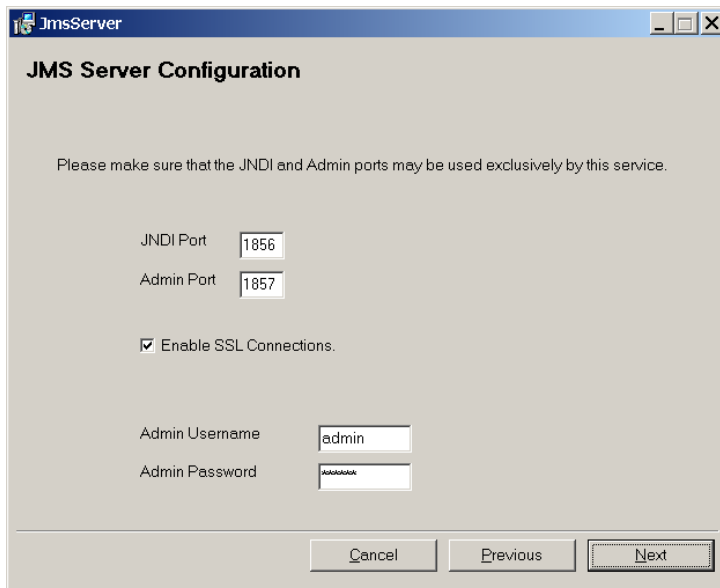
7. Click **Next** to continue.

The system displays the following screen:



8. Click **Next** to accept the default installation location, or click **Browse** and choose an alternate directory for the installation.

The system displays the following screen:

A screenshot of a Windows-style dialog box titled "JmsServer" with a standard icon on the left and minimize, maximize, and close buttons on the right. The main title is "JMS Server Configuration". Below the title is a note: "Please make sure that the JNDI and Admin ports may be used exclusively by this service." There are two text input fields: "JNDI Port" with the value "1856" and "Admin Port" with the value "1857". Below these is a checkbox labeled "Enable SSL Connections." which is checked. At the bottom, there are two more text input fields: "Admin Username" with the value "admin" and "Admin Password" with masked characters. At the very bottom are three buttons: "Cancel", "Previous", and "Next".

JmsServer

JMS Server Configuration

Please make sure that the JNDI and Admin ports may be used exclusively by this service.

JNDI Port 1856

Admin Port 1857

☒ Enable SSL Connections.

Admin Username admin

Admin Password ****

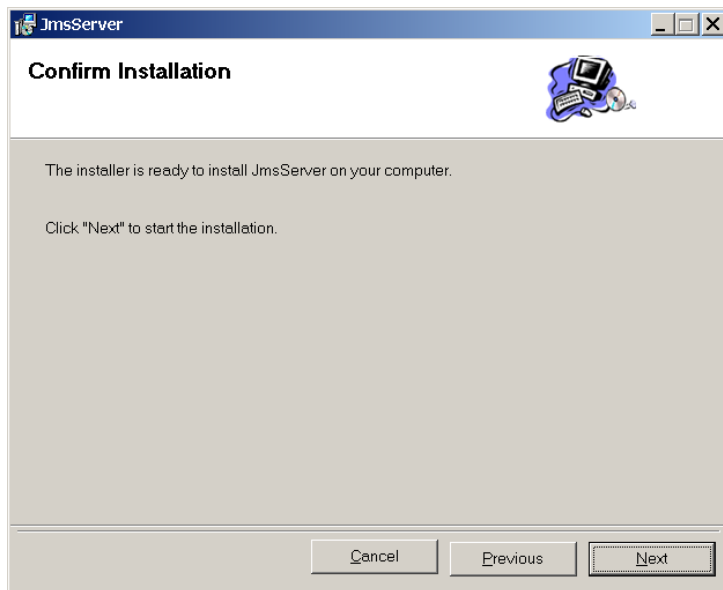
Cancel Previous Next

9. The Fiorano Settings are used for JMS traffic and administrative operations. By default these ports are set to 1856 and 1857, respectively. You may want to change these port settings if you have other applications using these ports.

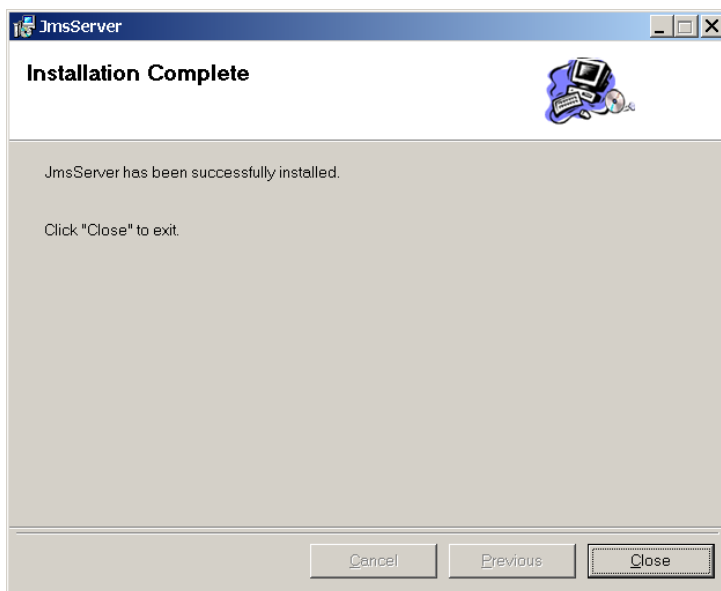
Note: It is recommended that you install Everserve to run Fiorano with SSL. Failure to do so may compromise the community against hacking, spoofing, or unauthorized access. All communities must be configured with all devices either installed to use SSL or not. With the exception of target only installed devices, Everserve does not allow mixed SSL/non-SSL devices within the same community. To switch devices between SSL and non-SSL, you must reinstall Everserve and choose the appropriate option.

10. Enter the Administrator account name and password.
11. Click **Next** to continue.

The system displays the following screen:



The system installs JMS on the device. After installation, the system displays the following screen:



12. Click **Close**.

Solaris JMS Only Installations

To install JMS only on a Solaris device:

Summary

1. Uninstall any prior versions of Everserve or JmsServer.
2. Install JMS using the JMS PKG installer.

The following sections provide instructions for each of the tasks described above.

Uninstalling Prior Versions of JmsServer

1. Login as root.
2. Uninstall any previous instance of JMS:

```
# pkgrm Jms
```
3. After the JmsServer has been uninstalled, delete the /JmsServer directory.

Note: If you wish to save any previous files in the /JmsServer directory, save them to a separate location.

```
# cd /opt/synchron  
# rm -rf JmsServer
```

Installing the Jms Server

1. Go to a temporary directory in which to begin the JmsServer installation. For example:

```
# cd /usr/tmp
```
2. Copy the **JmsServer.pkg.tar** file from the CD to the temporary directory.

```
# cp /cdrom/Installers/Solaris/JmsServer/JmsServer.pkg.tar .
```
3. Untar the JmsServer file:

```
# tar -xvf JmsServer.pkg.tar
```
4. Install JmsServer from the temporary directory:

```
# pkgadd -d .
```

The system will display a series of prompts required to complete the installation.

5. Upon completion, the following subdirectories are created in the JmsServer home directory (`/opt/synchron/JmsServer` in a default installation).

- `/bin`
- `/lib`
- `/scripts`
- `/jre`

6. Start the Jms Server.

```
# jmsserver start
```

Post Installation Tasks

After installing Everserve, ensure the following directories have been installed on the system (Windows systems installs into \Program Files\Synchron Networks\Everserve\, while UNIX systems installs into /usr/local/synchron/):

- bin
- docs
- external
- jre (if Everserve installed a Java Virtual Machine at time of installation)
- lib
- server
- _uninst

When running Everserve as a Windows Service (that is, Windows NT and Windows 2000), the Everserve installation program will create the following registry entries:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everserve
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everserve\Enum
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everserve\Parameters
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everserve\Security

When installing Everserve on devices that have access to the Web interface (that is, devices installed with community manager or publisher capabilities) the Everserve installation program will create the following additional registry entries:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everweb
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everweb\Enum
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everweb\Parameters
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Everweb\Security

System Configuration

Everserve's installation process will completely configure systems to run Everserve with little to no further configuration modifications necessary to create communities and begin delivering packages. However, there may be times when customizing certain configuration elements can improve performance of your Everserve network while meeting the specific delivery requirements of peers in the Everserve community.

This section provides information on specific configuration options that may be modified to meet your computing environment's needs. Instructions on how to perform these configuration operations are included in this section.

The topics in this section include:

- [Configuring Alternate TCP and JMS Ports](#)
- [Limiting Target Capabilities](#)
- [Configuring System Logs](#)
- [Setting-up Email Notification](#)
- [Setting Delivery Timeouts](#)

Configuring Alternate TCP and JMS Ports

At install time, Everserve messaging ports are designated for RMI TCP and Fiorano JMS transmissions to community peers. To ensure efficient and reliable transmissions, it is highly recommended that both the RMI TCP and the Fiorano JMS transmissions are set up on open ports not in use by any other applications or instances.

After installing Everserve, you can change the port designations for RMI and JMS transmissions. This is done by modifying the Everserve.xml configuration file and changing the port assignments as appropriate for message transmissions.

Configuring RMI TCP Port

Everserve uses port 1099 as the default TCP port for RMI transmissions for transmissions to community devices. If this port setting is currently in use by another application, it is highly recommended that a different port be used to handle Everserve activity.

The TCP port setting is specified when installing Everserve on the community manager system, in which case, Everserve automatically assigns port information in the appropriate configuration files. However, the RMI port can be changed at later time from 1099 to a port assonant not currently in use.

To Change the RMI Port Setting Used:

Edit the Everserve.xml configuration file located in the \Synchron Networks\Everserve\server\config directory and change the RMI port setting from 1099 to a port assignment not in use by another application.

The following example illustrates the string requiring modification:

```
<ServerSystemProperty property="java.naming.provider.url"
    type="string" value="rmi://localhost:1099"/>
```


Configuring Fiorano JMS Ports

Everserve uses port 1856 for Fiorano JMS messaging, and port 1857 for Fiorano administrative transactions. If either of these port settings are currently in use by other applications or other Fiorano instances, it is highly recommended that different ports be used to handle Everserve activity. This will help to ensure efficient and reliable package deliveries to all peers in the community.

The JMS port settings are specified when installing Everserve on the community manager system or JMS server located in the DMZ. However, these port settings can be changed after installation by modifying the Everserve.xml configuration file as described below.

To Change the Fiorano JMS Port Settings Used:

1. Stop Everserve on the device.
2. Edit the Everserve.xml configuration file located in the \Synchron Networks\Everserve\server\config directory and change the JMS transport settings.
For example, to change the Fiorano port settings from 1856 and 1857 to 1900 and 1901 respectively, edit the following string as shown below.

Change From:

```
<TransportConfigurationProperty
    property="java.naming.provider.url" type="string"
    value="http://localhost:1856"/>

<TransportConfigurationProperty
    property="admin.java.naming.provider.url" type="string"
    value="http://localhost:1857"/>
```

Change To:

```
<TransportConfigurationProperty
    property="java.naming.provider.url" type="string"
    value="http://localhost:1900"/>

<TransportConfigurationProperty
    property="admin.java.naming.provider.url" type="string"
    value="http://localhost:1901"/>
```

3. Save the changes made to Everserve.xml and restart Everserve.

Limiting Target Capabilities

Because Everserve can deploy files and applications to any device in an Everserve community, publishers have the power to read directories, and to overwrite files and applications without human interaction or permissions. There may be cases where you would not want to expose certain directories, files, or applications to a publisher that could extract directory information or overwrite files.

To limit a target's susceptibility to unwanted operations, you can control the target's exposure from a publisher as follows:

1. Set the target's Everserve service to run as a particular user. For example, you can set Everserve to run as "guest" so that when the user "guest" is logged in certain sensitive data is not accessible. This can be set at install time or after Everserve has been installed on the target. Ensure this account has the following accessibility rights:
 - The account is able to log on as services.
 - The account has FULL access to Everserve install directory.
 - The account has FULL access to the \Everserve\server directory.
 - The account has Read and Execute; Read; List rights to other directories (the directories you wish to lock down).
2. Ensure that your sensitive data is locked down so that the Everserve user (guest in this example) can not access it. Note that you can choose to allow packages to deliver new files into sensitive folders, without the right to read the files. You can lock down individual files or directories so the publisher does not have access to these directories and files. By locking down the appropriate executables, you can prevent scripts from executing those executables as well. For information on how to set the target's system to disallow certain access and operations from a publisher, see [Locking Down Directories, Files, and Executable Operations](#).

With such a setup, if the publisher attempts to deliver a package that accesses the sensitive data, the package status indicator will display a red light indicating a delivery failure, and the return receipt will indicate a stderr due to an "Access denied" error occurring on the target system.

Configuring Everserve to Run as a Specific User

During install time, you can choose to configure Everserve to run based on the account name supplied. However, if you have already installed Everserve to run on the local system regardless of the account name supplied, you can change this configuration option without having to reinstall Everserve. (Note that the following instructions are for Windows 2000 devices. Although the panel names may be different on other Windows OS devices, the process of setting permissions is similar in concept.)

To Configure Everserve on Windows Devices:

1. Open the Windows **Services** panel. Choose:
Start|Settings|Control Panel|Administrative Tools|**Services**
2. Right-click on the Everserve service and choose **Properties**.
3. Click the **Log On** tab.
4. Click the **This Account** radio button.
5. Fill in the Windows account information as appropriate.
6. Click **OK**.

Locking Down Directories, Files, and Executable Operations

There may be cases where you may want to protect certain files or directories from being overwritten on target systems. The following instructions describe how to lock down target systems so that specified directories and files can not be overwritten during package delivery and execution.

To Lock Down File Systems on Windows NT Devices:

1. Open the device's Explorer or My Computer panel to display a list of all folders on the device.
2. Right-click the directory or file you wish to lock down and choose **Properties**.
3. Click the **Security** tab.
4. **Add/Remove** users from the list and give them the appropriate permissions. Make sure the Everserve user **does not** have access to the file/directory for which you are setting security permissions, and does have access to the files and directories that you leave open with change permissions.

Note: You must ensure that the Everserve user has access to the \Synchron Networks directory. To do this, use the opposite procedure described in the lock down instructions in this section.

To Lock Down File Systems on Windows 2000 Devices:

1. Open the device's Explorer panel to display a list of all folders on the device.
2. Right-click the directory or file you wish to lock down and choose **Properties**.
3. Click the **Sharing** tab.
4. Click the **Share this folder** radio button.
5. Click **Permissions**.

The system displays a Name list box from which to select a user name, and a Permissions list box from which to set access rights to the folder or file currently selected.

6. Click **Add**.

The system displays a Name list box from which to select an account in which to grant or remove permissions to this folder or file.

7. Select an account name for which you wish to set permissions to this folder or file, then click **Add**. To select multiple accounts, press and hold the CTRL key while making selections.
8. Click **OK**.
9. Click an account from the Name list box, then click the privileges you wish to grant or remove from the list of Permissions check boxes.
10. Click **Apply**.
11. Repeat Steps 9 and 10 for all accounts for which to grant or restrict permissions for this folder or file.
12. Click **OK** to save the permission settings.
13. Click **Apply** on the Properties dialog box.
14. Click **OK** to save all changes.

Configuring System Logs

System log files are used to monitor and troubleshoot system activity and events. Everserve uses the Log4J.xml configuration file to specify which Everserve activities and events are logged, the amount of event detail to log, and the location in which to store log files. By default, logs are initially set to record activity and events that meet or exceed a “warning” state of operation, and are written to the `\Synchron Networks\Everserve\server` directory. The operational state at which logs are created, the activity and event detail recorded, and the default logging directory can be changed by modifying the Log4J.xml configuration file located in the `\Synchron Networks\Everserve\server\config` directory.

Changing Logging Layout and Format

Log4J.xml uses conversion characters to specify which events are logged, and the format in which information is displayed. The following table lists the conversion characters used to specify logging properties in the Log4J.xml configuration file. Use the conversion characters and formats shown in the following table to customize specific logging results for the device.

Character	Description
%c	Category of the logging event.
%C	Fully qualified class name of the caller.
%d	Date of the logging event.
%F	File name where the logging request was issued (caution: extremely slow).
%l	Location information of the caller (caution: extremely slow).
%L	Line number from where the logging request was issued (caution: extremely slow).
%m	Application-supplied message.
%M	Method name from where the logging request was issued (caution: extremely slow).
%n	Line separator.
%p	Priority of the logging event.
%r	Number of milliseconds since the start of the application.
%t	Name of the thread that generated the logging event.
%x	Nested diagnostic context associated with the thread.
%%	A single percent sign.

You can customize how information in the log files is displayed by specifying the format modifiers as shown in the following table:

Format Modifier Example	Description
%20c	Left pad with spaces if category is less than 20 characters long.
%-20c	Right pad with spaces if category is less than 20 characters long.
%.30c	Truncate from the beginning if category is more than 30 characters long.
%20.30c	Left pad 20 chars + truncate from beginning if more than 30 characters long.
%-20.30c	Right pad 20 chars + truncate from beginning if more than 30 characters long.

Comment Tags

By default, several sections of Log4J.xml are commented-out to reduce the amount of logging for specific activities or events. Comment tags begin with `<!--` and end with `-->`. To un-comment a section, remove both beginning and ending comment tags from the section; conversely, to comment-out a section, add beginning and ending comment tags to the section as appropriate.

The following sections describe how to modify the Log4J.xml configuration file for:

- [Specifying Logging Priority Values](#)
- [Specifying Format and Destination of STDOUT Logging](#)
- [Specifying Logging Priority Values](#)
- [Logging Java Exceptions](#)
- [Logging Everserve Start and Stop Activity](#)
- [Logging Server Connectivity Problems](#)

Specifying Logging Priority Values

Priority values are used to specify which activities and events to log based on event severity. This entry defines the logging behavior for Everserve activities and events only—it does not log third-party or operating system events.

Example:

```
<category name="com.synchronnetworks" additivity="false">

    <priority value="warn" />

    <appender-ref ref="STDOUT" />

</category>
```

Note: If the “priority value” section is commented out, it will effectively disable all Asserts in Everserve (which use the “fatal” level). You can selectively enable logging for specific areas of Everserve by un-commenting any of the categories described in the following sections.

Priority order is DEBUG < INFO < WARN < ERROR < FATAL. These levels of logging are as follows:

Priority Value	Description
Fatal	Enables Asserts in all Everserve code.
Error	Enables only error level messages (few).
Warn	Enables only warning level messages (more, still few).
Info	Enables info level logging (nearly none).
Debug	Enables debug logging for entire Everserve product, all classes. This is the most verbose setting and will show debug tracing for every single class enabled for Log4J.

Note: If a single log4J message prints multiple times it means this message is enabled by multiple areas. Be sure to add the ‘additivity=“false”’ modifier to all such categories to avoid this behavior.

Specifying Format and Destination of STDOUT Logging

You can change the format and destination for all log entries written for STDOUT for all Everserve log messages. To change STDOUT logging properties, change the value string for the "param name" tag (shown below) in the Log4J.xml configuration file. Refer to [Changing Logging Layout and Format](#) for descriptions and examples of formatting strings used to alter default system logging behavior.

Example:

```
<appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
            value="%d %-5p [%t] - %m\r\n"/>
    </layout>
</appender>
```

Specifying Format and Destination of STDERR Logging

To change the format and destination for all log entries written to STDERR, modify the value in the "param name" tag (shown below). Modifying this parameter will change the format of all Everserve error log messages. Refer to [Changing Logging Layout and Format](#) for descriptions and examples of formatting strings used to alter default system logging behavior.

Example:

```
<appender name="STDERR" class="org.apache.log4j.ConsoleAppender" >
    <param name="target" value="System.err"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
            value="%d %-5p [%t] - %m\r\n"/>
    </layout>
</appender>
```

Logging Java Exceptions

Enable (un-comment) the `com.synchronnetworks.util.EverOnException` category of logging to further investigate a Java exception that Everserve is throwing. Note that not all exceptions are fatal as Everserve has built-in recovery code to handle many exceptions.

Example:

```
<category name="com.synchronnetworks.util.EverOnException"
  additivity="false">

  <priority value="error" />

  <appender-ref ref="STDOUT" />

</category>
```

Logging Everserve Start and Stop Activity

Enable (un-comment) the `com.synchronnetworks.nile` category of logging to investigate why the Everserve device does not start or why it seemed to stop unexpectedly.

Example:

```
<category name="com.synchronnetworks.nile"  additivity="false">

  <priority value="debug" />

  <appender-ref ref="STDOUT" />

</category>
```

Logging Server Connectivity Problems

Enable (un-comment) the `com.synchronnetworks.roles` category of logging to investigate difficulties in one Everserve server connecting to another.

Example:

```
<category name="com.synchronnetworks.roles" additivity="false">
    <priority value="debug" />
    <appender-ref ref="STDOUT" />
</category>
```

If you need further information after enabling `com.synchronnetworks.roles` (described above), enable the `com.synchronnetworks.messenger` category of logging. This area is very verbose; use with caution when making deliveries.

Example:

```
<category name="com.synchronnetworks.messenger" additivity="false">
    <priority value="debug" />
    <appender-ref ref="STDOUT" />
</category>
```

Setting-up Email Notification

Everserve supports the ability to automatically send an email notification based on system and delivery events. Email notification is specified in the Everserve.properties file. By default, this feature is turned off.

To Set-up Email Notification:

1. Using a text editor, open Everserve.properties located in \Synchron Networks\Everserve\server\config.
`isEnabled=true`
2. Change the **isEnabled** parameter to TRUE. For example:
`isEnabled=true`
3. Enter a known valid SMTP address for the **smtpHost** parameter. For example:
`smtpHost=my.example.com`
4. Enter a known valid SMTP email address for the **from** parameter, For example:
`from=Everserve@my.example.com`
5. Enter a valid email address in which to send the email notification in the **to** field. For example:
`to="NetworkAdmin@my.example.com"`
6. Change the **notifyOnFailedReceipt=false** parameter to TRUE. For example:
`notifyOnFailedReceipt=true`

The following example shows how Everserve.properties might be modified to set up email notification:

```
# =====
# ===== EMAIL NOTIFICATION CONFIGURATION =====
# = Since this section is optional, it will expand in =
# = the output only if it already exists in the XML   =
# = or the enabled property below is true            =
# =====
#EverserveConfiguration.EmailNotificationConfiguration.isEnabled=true
#EverserveConfiguration.EmailNotificationConfiguration.smtpHost=my.example.com
#EverserveConfiguration.EmailNotificationConfiguration.from=Everserve@my.example.com
#EverserveConfiguration.EmailNotificationConfiguration.to=Admin@my.example.com
#EverserveConfiguration.EmailNotificationConfiguration.notifyOnFailedReceipt=true
```

7. Stop Everserve on the device, and run **xs1t.bat** from the `\Synchron Networks\Everserve\server\config` directory as follows:
`C:\.\Synchron Networks\Everserve\server\config> xs1t.bat.`
8. Restart Everserve to enable email notification.

Setting Delivery Timeouts

Delivery timeout is set in Everserve.xml on the target. By default, deliveries are set to timeout at 43,200 seconds (12 hours), at which time the delivery fails and a return receipt is sent to the publisher that originated the delivery. The status of the delivery changes from "Pending" to "Fail," with details of the error listed in the return receipt.

The default delivery timeout (43,200 seconds) can be changed by modifying the Everserve.xml configuration file located in \Synchron Networks\Everserve\server\config.

To Change Delivery Timeouts:

1. Stop Everserve on the target device.
2. Using an XML or text editor, open Everserve.xml located in \Synchron Networks\Everserve\server\config.
3. Go to the `TransportConfigurations` section and change the value (entered in seconds) for `defaultMaxRunTime` from 43200 to a new value.

For example, to have deliveries timeout after 2 minutes, set the `TransportConfigurations` value to 120. For example:

Change From:

```
<TransportConfigurations defaultTimeout="300"
    defaultMaxRunTime="43200" defaultBlocksize="51200" maximumRe-
    ceiveBufferSize="0"/>
```

Change To:

```
<TransportConfigurations defaultTimeout="300"
    defaultMaxRunTime="120" defaultBlocksize="51200" maximumReceive-
    BufferSize="0"/>
```

4. Save the file and restart Everserve.

Part 2

Community Management

Command Line Interface Operations

The task of creating and managing a community is the responsibility of the community manager. The community manager creates and maintains a community's definition, which includes creating the community, creating peers, and populating the community with peer definitions.

This topics in this section include:

- [Using the Interactive Command Shell](#)
- [Controlling Operative States](#)
- [Managing Communities](#)
- [Managing Peers](#)
- [Publishing Commands](#)
- [Information Services Commands](#)

Using the Interactive Command Shell

Everserve provides a command shell from which to enter Everserve commands. The shell eliminates the need to enter “everserve” before each command entered, thus reducing typing errors and improving performance.

To enter the command shell, simply type “Everserve” at the command prompt. For example:

```
C:\> everserve
```

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All  
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

```
Everserve>
```

For illustration purposes, the instructions and examples that follow **do not** use the interactive shell; therefore, all commands are preceded with “everserve.”

Controlling Operative States

The following sections describe the various commands used to control Everserve operational states. For Solaris systems, you must be logged on as root, or logged into a specific user account that was enabled at install time with rights to stop, resume, and pause an Everserve service.

Note: The issuer of each command listed must be logged in to the device as a root or Administrator.

Starting Everserve

The **start** command starts an Everserve process. For Windows devices, Everserve is automatically started upon system reboot. In cases where Everserve was stopped, it can be started again from the command line. For Solaris devices, Everserve is manually started from the command line.

To start an Everserve process, enter the following command:

Command Syntax:

```
$ everserve start
```

Stopping Everserve

The **stop** command shuts down the Everserve process, including all other required processes on the device. Stopping an Everserve process on a target will not interfere with message delivery since messages sent to the target are held in the sender's JMS queue until the Everserve process is resumed on the target. However, stopping an Everserve process on a publisher or relay device will stop the JMS queue running on that device. When the JMS queue is stopped, no messages can be sent to targets and return receipts will not be sent to the originating publisher until Everserve is restarted.

Command Syntax:

```
$ everserve stop
```

Pausing Everserve

The ***pause*** command suspends all Everserve processes until the resume command is given. When paused, no other Everserve commands (except *resume*), including package delivery can be executed. Packages sent to a paused server will be maintained in queue by the publisher so that once the server is resumed (or stopped and started again), any packages it missed while it was paused will then be received.

Command Syntax:

```
$ everserve pause
```

Resuming Everserve

The ***resume*** command re-instates Everserve processes to a fully operational state for all Everserve commands.

Command Syntax:

```
$ everserve resume
```

Managing Communities

Managing communities involves the creation and maintenance of the community definitions. Each community is created and populated with peers. When a peer joins a community, it receives information on its role in the community, routing information that it needs to know about, and a trust model is established between the peers and the community manager. As a network topology changes, it becomes increasingly important to maintain and update a community's definition to reflect these topology and/or routing changes.

The following sections describe the operations and tasks performed to create and maintain community definitions.

Community Descriptions

A community description is a descriptive label bound to a community. Providing a description for the community can be very helpful in differentiating between multiple communities. It is important to note the following criteria regarding community descriptions:

- If no community description is provided, Everserve will omit the community description from the community definition. Therefore, if you are managing multiple communities, it is strongly recommended that you provide descriptive text that you can use to distinguish one community from another.
- If a multi-word community description is provided without enclosing quotation marks, only the first word of the description after the community name will be saved as the descriptive name for the community.
- If you change the community name, make sure you also change the description to reflect this change. Otherwise the old description will be associated with the new community name (see [Changing a Community Definition](#)).

Creating a New Community

The **create community** command is used to create a new community, and to create and add the community manager peer to the community. After creating the community, the community manager then creates peers and adds the peers to the community definition. The following example illustrates the syntax used to create a new community:

Command Syntax:

```
everserve create community {-c <communityName>}
                        {-p <communityManagerName>} [-d <"communityDescription">]
```

Parameters and Options:

Parameter	Required	Description
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
{-p <communityManagerName>}	Y	The peer name you wish to assign as the community manager. If a peer with this name already exists, the system adds the peer to the community with the role of community manager. If this peer does not exist, the system creates the peer with the role of community manager, and automatically adds and joins the peer to the community.
[-d <"communityDescription">]	N	Descriptive name for the community. Community descriptions are alphanumeric and may include symbols. Spaces can be used in the description provided the description is enclosed in quotation marks. If no community description is provided, Everserve will omit the community description from the community definition.

Note: Windows operating systems reserves the nomenclatures "com<x>" and "lpt<x>" for device designation (where <x> is an integer ranging from 0-9). An error will be returned if you attempt to create a community using either of these device naming designators.

Example:

The following example illustrates creating the community, “*HR*,” with a community description of “*Human Resources*,” and the community manager peer, “*HRcm*.”

```
$ everserve create community -c HR -d "Human Resources" -p HRcm
```

Changing a Community Definition

The ***change community*** command is used to change the name or description of an existing community.

Command Syntax:

```
everserve change community {-c <communityName>}  
[-n <NewcommunityName>] [-d <"NewCommunityDescription">]
```

Parameters and Options:

Parameter	Required	Description
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
[-n <NewcommunityName>]	N	New descriptive name for the community. Community names are alphanumeric and may include symbols. Community names are case sensitive.
[-d <"NewCommunityDescription">]	N	New description for the community. Community descriptions are alphanumeric and may include symbols. Spaces can be used in the description provided the description is enclosed in quotation marks.

Example:

The following example illustrates changing the name for the community, *HR*, to “*HumanResources*,” with the new community description of “*Regional Human Resources Headquarters*.”

```
$ everserve change community -c HR -n HumanResources -d "Regional Human  
Resources Headquarters"
```

Deleting a Community

The **`delete community`** command is used to permanently remove the community definition from your Everserve network. The community definition specifies which server will publish to the community, which servers will act as community relays to fan out messages to targets, and which targets will receive packages. Because the **`delete community`** command removes only the community definition, individual peer definitions remain intact and can be added to other communities as needed. The following example illustrates the syntax used to delete a community:

Note: Deleting a community does not delete any peer definitions, rather, the peers remain intact but become orphaned and unable to receive packages until they are added to, and join a new community. This predication also includes the community manager peer, that is, you can delete a community, but the community manager peer will retain its description that it is a manager for that community. Therefore, when deleting a community, it is recommended that you either delete the community manager peer after deleting the community (if it is not a community manager for any other community), or change the community manager peer's description to indicate that it is no longer a community manager. See [Changing a Peer Definition](#) or [Deleting a Peer](#) for additional information.

Command Syntax:

```
everserve delete community {-c <communityName>}
```

Parameters and Options:

Parameter	Required	Description
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.

Example:

The following example illustrates deleting the community, "HR:"

```
$ everserve delete community -c HR
```


Recreating a Community Seed

The **create seed** command is used by the community manager to generate a community seed file. The seed file is automatically generated when the community and community manager are created. However, in cases where the community definition changes (such as when changes to the community name, description, or community manager are made), the seed file can be generated again and distributed to all new Everserve devices wishing to join the community.

The seed file contains the community manager's public key used to validate device identities for those devices wishing to join the community and receive packages. Once the seed has been created, the seed must then be copied to all devices in the community. Using the *join* command, the device then initiates a challenge-response protocol with the community manager.

Note: The community seed file must be copied to the `\Synchron Networks\Everserve\server` directory on each device wishing to join the community.

Command Syntax:

```
everserve create seed {-c <communityName>}
```

Parameters and Options:

Syntax	Required	Description
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.

Example:

The following example illustrates creating a community seed for the community "HR."

```
$ everserve create seed -c HR
```

Managing Peers

From the command line, you can create peers, add and remove peers from a community, and change and delete peer definitions. The following sections describe how to manage the peer definitions in your communities.

Peer Descriptions

Providing a description for each peer can be very helpful in differentiating between peers in a community. It is important to note the following criteria regarding peer descriptions:

- If no peer description is provided, Everserve will omit the peer description from the peer definition. Therefore, if you have multiple peers with similar peer names or designations in your communities, it is strongly recommended that you provide descriptive text that you can use to distinguish one peer from another.
- If a multi-word peer description is provided without enclosing quotation marks, only the first word of the description after the peer name will be saved as the descriptive text for the peer.
- If you change the peer name, make sure you also change the description to reflect this change. Otherwise the old description will be associated with the new peer name.

Creating a Peer

The **create peer** command is used to create a new peer. Newly created peers are not initially associated with a community until they are added to a community and given a role in the community. Peers are not entitled to receive packages until they join the community to which they were added.

Command Syntax:

```
everserve create peer {-p <peerName>} {-h <hostName>}  
[-d <"peerDescription">]
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name for the peer. Peer name strings are case sensitive and alphanumeric.
{-h <hostname>}	Y	The actual location of the machine (peer) on the network. The hostname specified can be a fully qualified hostname, or an IP address. The following are examples of valid hostnames: <ul style="list-style-type: none">• MyMachine• MyMachine.example.com• 112.207.77.22
[-d <"peerDescription">]	N	Descriptive name for the peer. Peer descriptions are alphanumeric. Spaces can be used in the description provided if the description is enclosed in quotation marks. If no peer description is provided, Everserve will omit the peer description from the community definition. If a peer description is provided without the enclosing quotation marks, only the first word of the description after the peer name will be saved as the descriptive text for the peer.

Example:

The following example illustrates creating a peer with the peer name of "*Region2*," located on the server "*MySystem.example.com*," with a peer description of "*Northwest HR Region Server*."

```
$ everserve create peer -p Region2 -h MySystem.example.com  
-d "Northwest HR Region Server"
```

Adding Peers to a Community

The **add peer** command is used to add an existing peer definition to a community, to assign the peer a role in the community, and to specify from which publisher or relay the peer should receive packages. Unless otherwise specified, all peers added to a community will have a default role of “target.” Supported peer roles include: community manager, publisher, relay, and target.

Command Syntax:

```
everserve add peer {-p <peerName>} {-c <communityName>} [-r <role>]
[-s <sender...>]
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name of the peer you wish to add to the community. Peer name strings are case sensitive and alphanumeric.
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
[-r <role>]	N	Peers may have one role within the context of a community. Peer roles include: <ul style="list-style-type: none"> • CM (community manager) • Pub (publisher) • Relay • Target If no role is specified, the peer will be assigned the default role of target.
[-s <sender...>]	N	Peer name of the sender, either the publisher or relay sending the package. If no sender is specified, the peer will receive packages from all community publishers. See Default Peer Connections (When a Sender is Not Specified) for information on peer connection rules. <p>Multiple senders may be specified for a peer by separating each peer name with a space, for example:</p> -s sender1 sender2 sender3

Note: Although it is possible to add a peer to a community and assign it the role of publisher, community manager, or relay, the peer will only function as a target unless the corresponding role capabilities were installed on the peer device. For example, a device assigned the role of publisher must have received an installation that included publisher capabilities.

Example:

The following example illustrates adding the peer “*PacRim Region*” to the community “*HR*.” Note that the role for this peer in the community is a relay which is connected to two publishers in the *HR* community, “*HRpublisher1*” and “*HRPublisher2*.” Both publishers will then be able to send packages to the relay.

```
$ everserve add peer -p "PacRim Region" -c HR -r relay -s HRpublisher1  
HRpublisher2
```

Specifying Peer Connections in the Community

Peer connection is the path by which two or more peers communicate with each other. A peer that is connected to a publisher communicates with that publisher in the form of receiving packages and sending return receipts. A peer that is connected to a relay operates in the same manner - receiving packages that pass through the relay from the publisher (or perhaps other relays), and sending return receipts through the relay (or other relays) to the publisher that originated the package.

In this manner, all communities are configured in a single path hierarchical manner, that is, all targets are reachable by all publishers, even when one or more relays are used in this singular path to complete the connections between publishers and targets.

By default, if you are adding a target or a relay to the community and do not specify a sender, Everserve will automatically connect the target or relay to all publishers in the community. This ensures that there are no orphaned targets or relays in the community that cannot receive packages. Ideally, a target or relay should be connected to either all publishers, or one or more relays in the community.

Connections and disconnections are made between peers using the “-s” parameter while adding a peer or removing a peer from a community, adding senders to the peer, or when changing a peer’s definition.

There are several rules that must be adhered to when making connections to sender peers, and when disconnecting from sender peers.

Connecting a Sender to an Everserve Device

The **add sender** command is used to connect a device to a sender (either all publishers or one or more relays). By default, all devices are connected to all publishers in the community unless otherwise specified. In most cases, the sender is connected to the device when the peer is added to the community. You can, however, connect a device to multiple senders at any time using the **add sender** command.

Command Syntax:

```
everserve add sender {-p <peerName>} {-c <communityName>}
                    {-s <sender...>}
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name of the peer you wish to add to the community. Peer name strings are case sensitive and alphanumeric.
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
{-s <sender...>}	Y	Peer name of the sender, either the publisher or relay from which packages will be received. If no sender is specified, the peer will receive packages from all community publishers. See Default Peer Connections (When a Sender is Not Specified) for information on peer connection rules. Multiple senders may be specified for a peer by separating each peer name with a space, for example: -s sender1 sender2 sender3

Example:

The following example illustrates using the **add sender** command to connect the target “*PacRim Region*” in the community “*HR*” to the publisher “*HRpublisher*.”

```
$ everserve add sender -p "PacRim Region" -c HR -s HRpublisher
```

Connection Rules

There are several rules that must be considered when making connections to sender peers. The following sections describe these rules:

- [Default Peer Connections \(When a Sender is Not Specified\)](#)
- [Connections Made by Specifying a List of Senders](#)
- [Connecting Senders to Relays and Targets](#)
- [Changing the Sender Connections for a Peer](#)
- [Removing a Sender from a Peer](#)

Default Peer Connections (When a Sender is Not Specified)

The following sections describe how Everserve automatically handles peer connections if no sender is specified (by **not** using the “-s” parameter) when adding a peer to a community.

Connecting Peers to a Publisher

- If there are **no relays** in the community, all existing targets, if any, are connected to *all* publishers in the community.
- If there **are relays** in the community, all **first level relays**, which are relays that receive packages from publishers only, are connected to the publisher.

Connecting a Relay to a Sender

When the community already has one or more publishers:

- If there are **no relays** already in the community, the relay is connected to *all* publishers in the community.
- If there **are relays**, Everserve prompts you to explicitly define the desired connections such that the relay is either connected to *all* publishers, or to one or more other relays.

When the community does not have any publishers:

- If there are **no relays** in the community, all targets, if any, are connected to the relay being added to the community.
- If there **are relays** already in the community, all first level relays are connected to this new relay. The newly added relay becomes the first level relay; all other relays already in the community drop down one level in their connection status, and automatically connect to this new relay. That is, a relay that was a first level relay now becomes a second level relay. A relay that was a second level relay becomes a third level relay, and so on so that the single path hierarchical tree is maintained with the newest relay always being added at the top (first level) of the tree.

Specifying Peer Connections

The following sections describe how to specify sender connections for peers using the “-s” parameter.

Connecting a Target to a Sender

The information in this section describes how Everserve handles connections between peers when a sender is specified (using the “-s” parameter) when adding a peer to a community.

- If there are **no relays** in the community, the target is connected to the publisher(s) specified.
- If there **are relays** in the community, Everserve prompts you to explicitly define the desired connections such that the target is connected either to *all* publishers or to one or more relays.

Connections Made by Specifying a List of Senders

The information in this section describes how Everserve handles connections between peers when multiple senders are specified (using the “-s” parameter followed by a list of sender peers) when adding a peer to a community.

Note: Because publishers are the top level senders in a community, the “-s” parameter is not valid when adding a publisher to a community.

- When a **relay** is added, either *all* publishers or one or more relays are listed with the “-s” parameter. Note that when specifying one or more relays, the relay you are adding will not be a first level relay, rather it will be positioned one level down from the relay specified.
- When a **target** is added, either *all* publishers or one or more relays are listed in the “-s” parameter.

Connecting Senders to Relays and Targets

The information in this section describes the rules for connecting senders (that is, publishers and relays) to relays and targets in a community. In general, relays and targets may be connected to either *all* publishers in a community, or to one or more relays in a community. *Relays and targets may not be connected to both publishers and relays in the same community.*

Note: Because publishers are the top level senders in a community, the “-s” parameter is not valid when adding a publisher to a community.

Adding Senders to a Relay

When you add senders to a relay, you should always maintain a single path of communication between senders and receivers to ensure no peers in the community become orphaned and unable to receive packages. The rules for adding senders to a relay are as follows:

- If you want to connect a publisher to a relay, you must connect *all* publishers in the community to the relay. Attempting to connect a relay to only one publisher of many in a community, or to both relays and publishers at the same time will result in an error.
- If you want to connect the relay to another relay, you must connect it to one or more relays in the community. Attempting to connect a relay to both relays and publishers will result in an error.
- If the relay is not connected to any other peer, you may connect the relay to either *all* publishers, or to one or more relays.

Adding Senders to Targets

The rules for adding senders to targets are similar to that of adding senders to relays; targets must be connected to either *all* publishers in a community, or to *one or more* relays in a community, but not connected to both publishers and relays in the same community. For example:

- If you want to connect a target to a publisher, it must be connected to *all* publishers in the community. An error will result if you attempt to connect the target to only one publisher of many in a community, or to both relays and publishers at the same time.

However, if a publisher is added to a community, and the target is connected to *all* existing publishers in the community, you can add the new publisher as a sender for the target (see [Changing the Sender Connections for a Peer](#) for additional information).

- If you want to connect the target to a relay, you must connect it to *one or more* relays in the community. An error will result if you attempt to connect the target to both relays and publishers at the same time.

However, if a relay is added to a community, and the target is connected to *one or more* relays in the community, you can add the new relay as a sender for the target (see [Changing the Sender Connections for a Peer](#) for additional information).

Creating and Adding Several Peers to a Community

In cases where many peers (such as hundreds or thousands of peers) need to be added to the community, you can create a file containing a list of Everserve commands to create the peers and add each peer to the community definition. Once the file is created, you can use the **run** command to execute each Everserve command in the file.

Note: Everserve will administer each command in the file in the order in which it is listed. Therefore, when creating and adding peers using this method, you must always create a peer before adding a peer to a community.

1. Create a text file and create a list of peers you wish to create. For each peer you wish to add to the community, list the peer's hostname (-h), and a description for the peer (-d) using the **create peer** command and syntax. For example:

```
create peer -p CorpRelay -h corpServer -d "Corporate Relay"
create peer -p NWsales -h NWserver -d "Seattle Sales Office"
create peer -p SFsales -h SFserver -d "San Fran Sales Office"
create peer -p NEsales -h NEserver -d "Boston Sales Office"
create peer -p SWsales -h SWserver -d "New Mexico Sales Office"
```

2. After entering all **create peer** commands, use the **add peer** command to add each peer (-p) to the community (-c). When adding each peer, specify the role (-r) the peer will have in the community, and to which sender (-s) it will be connected. For example:

```
add peer -p CorpRelay -c SalesComm -r relay -s SalesComPublisher
add peer -p NWsales -c SalesComm -r target -s CorpRelay
add peer -p SFsales -c SalesComm -r target -s CorpRelay
add peer -p NEsales -c SalesComm -r target -s CorpRelay
add peer -p SWsales -c SalesComm -r target -s CorpRelay
```

In the example above, the relay, *CorpRelay*, is connected to—and will receive packages from—the community publisher, *SalesComPublisher*. All other peers (*NWsales*, *SFsales*, *NEsales*, *SWsales*) will act as targets and are connected to—and will receive packages from—the relay, *CorpRelay*.

Note: If no sender is specified, the peer will be automatically connected to all publishers in the community if no relays exist. If relays are present in the community, the peer will be connected to the level one relay. For specific details and rules regarding peer connections, see the section [Default Peer Connections \(When a Sender is Not Specified\)](#).

3. Save the file with an `.es` file extension.

The following example illustrates the entire file after all ***create peer*** and ***add peer*** commands have been listed:

```
create peer -p CorpRelay -h corpServer -d "Corporate Relay"
create peer -p NWSales -h NWserver -d "Seattle Sales Office"
create peer -p SFSales -h SFserver -d "San Fran Sales Office"
create peer -p NESales -h NEserver -d "Boston Sales Office"
create peer -p SWSales -h SWserver -d "New Mexico Sales Office"
add peer -p CorpRelay -c SalesComm -r relay -s SalesComPublisher
add peer -p NWSales -c SalesComm -r target -s CorpRelay
add peer -p SFSales -c SalesComm -r target -s CorpRelay
add peer -p NESales -c SalesComm -r target -s CorpRelay
add peer -p SWSales -c SalesComm -r target -s CorpRelay
```

4. Use the ***run*** command to execute the commands listed in the file, for example:

```
$ everserve run -f <exampleFile.es>
```

Everserve will open the file, and execute each command in the order listed.

Changing a Peer Definition

The ***change peer*** command is used to change a peer's definition or to change its definition in the community. With exception to the peer name and community name, all peer definition variables are optional. Any variable that is not included in the change peer syntax remains unchanged. Each of these scenarios are described below.

Note: Everserve does not permit changing a peer's hostname. If the hostname for a peer requires modification, you must delete the peer's definition, create a new peer definition, then add the new peer definition to the community. See [Deleting a Peer](#) for additional information.

Command Syntax for Changing the Peer Definition Only:

```
everserve change peer {-p <peerName>} [-n <NewpeerName>]
                    [-d <NewDescription>]
```

Command Syntax for Changing the Peer Definition in the Community:

```
everserve change peer {-p <peerName>} {-c <communityName>}
                    [-r <role>] [-s <sender...>]
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name of the peer you wish to change. Peer name strings are case sensitive and alphanumeric.
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
[-n <NewpeerName>]	N	New name for the peer.
[-d <"peerDescription">]	N	New description for the peer.

Syntax	Required	Description
[-r <role>]	N	Peers may have one role within the context of a community. Peer roles include: <ul style="list-style-type: none"> • CM (community manager) • Pub (publisher) • Relay • Target
[-s <sender...>]	N	Peer name of the sender, either the publisher or relay from which packages will be received. If no sender is specified, the peer will receive packages from all community publishers. See Default Peer Connections (When a Sender is Not Specified) for information on peer connection rules. Multiple senders may be specified for a peer by separating each peer name with a space, for example: -s sender1 sender2 sender3

Example: Changing a Peer Definition

The following example illustrates changing the peer name from “*Region2*” to “*NWoffice*.”

```
$ everserve change peer -p Region2 -n NWoffice
```

Example: Changing a Peer Definition in a Community

The following example illustrates changing the role of the peer “*Region2*” in the community “*HR*” to “*Relay*,” and connects the peer to the community publisher, “*HRpublisher*.”

```
$ everserve change peer -p Region2 -c HR -r relay -s HRpublisher
```

Changing the Sender Connections for a Peer

You can change the senders to which peers are connected using the "change peer -s" command. When you change a peer's senders, only those senders that are specified with the "-s" parameter will be connected to the peer. All senders previously connected to the peer will no longer be senders for that peer (unless they are defined as a sender when using the "-s" parameter).

Note: Because publishers do not have senders, the "-s" parameter used with the "change peer" command is not valid for publishers.

- If you want to change the sender connections for a relay, the new senders must be either *all* publishers, or one or more relays in the community.
- If you want to change the senders for a target, the new senders must be either *all* publishers, or one or more relays in the community.

Removing a Sender from a Peer

This section describes the affects of removing senders from peers (when using the "remove sender -s" command).

Note: Because publishers do not have senders, the remove sender command is not valid for publishers.

Removing a Sender From a Relay

- If the sender you want to remove is a first level relay, that is, a relay that is connected to publishers, an error will result indicating that publisher senders cannot be removed.
- If the sender is **not** a first level relay, but is connected to other relays, you can remove any of these relays in the community except the last one.

Removing Senders From a Target

- If the target is **not** connected to any publishers, that is, the target is connect to relays, you can disconnect the target from any relay sender in the community except the last one.

Note: If you attempt to disconnect a target from a publisher, an error will result indicating that publisher senders cannot be removed.

Command Syntax:

```
everserve remove sender {-p <peerName>} {-c <communityName>}
                        {-s <sender...>}
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name of the peer you wish to disconnect from the sender. Peer name strings are case sensitive and alphanumeric.
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.
{-s <sender...>}	Y	Peer name of the sender, either the publisher or relay from which packages are sent to targets. Multiple senders may be disconnected from the peer, in which case, enter each sender separating the sender peer name by a space. For example: -s sender1 sender2 sender3 Note: Everserve <i>does not allow all senders to be removed from the peer; a peer must be connected to at least one relay or all publishers in the community.</i>

Example:

The following example illustrates disconnecting the peer “NWoffice” from the sender “NWofficeRelay” in the community “HR.”

```
$ everserve remove sender -p NWoffice -c HR -s NWOfficeRelay
```

Activating a New Everserve Device

After the community and community manager have been created, the community seed file is automatically generated. This seed file must be copied to the `\Synchron Networks\Everserve\server` directory on each Everserve device in the community. Once the community seed has been copied to the device it must join the community to become an active and trusted community member.

When the **join** command is issued, the seed file (`<communityName.zip>`) is opened and a request message is sent to the community manager.

*Note: Because the seed file is a compressed **zip** file, the device wishing to join the community must have the ability to unzip the seed file to join the community.*

The **join** command is issued on each peer device as follows:

```
$ everserve join {-c <communityName>}
```

For example, to join the community called "News:"

```
$ everserve join -c News
```

Using the information in the seed file, Everserve establishes a secure communication channel. Creating a secure channel allows the new device to validate the identity of the community manager using certificate verification protocols. If the identity is not validated, the device aborts the process and logs an error message with the community manager and on its own local file system.

If the secure channel is successfully created, the community manager and device engage in a challenge-response protocol. The purpose of this protocol is to allow the community manager to trust the device. The newly joined device generates an unsigned certificate for itself and sends it to the community manager. The community manager signs the certificate, annotates the certificate with the role of that device within the community, keeps a copy of the signed certificate, then sends the certificate back to the new device. At this point, the device has joined the community and is considered an active and trusted member.

The community manager then sends a subset of the community specification to the new device, giving it the information it must have to function within the community. The device then saves this subset definition of the community on its local file system.

If the device is a target, it connects to the queue of the sender that was specified when the target was added to the community. The sender's queue for a target will be either all publishers in the community, or one or more relay queue(s).

Removing a Peer from a Community

The **remove peer** command is used to remove a peer from a community without losing any of the peer's definition. Once a peer has been removed from a community, appropriate information and notification is sent to all other peers in the community so that they may continue to function properly. For example, publishers and relays cannot publish to peers that have been removed from the community. Peers can be added and removed from communities repeatedly.

If a relay is removed from a community, any targets and other relays that were connected to this relay are automatically connected to the sender(s) of the removed relay. An exception to this rule is when the last or only relay in a community is removed. In this case, all remaining targets will automatically be connected to all publishers in the community).

Note the following connections rules:

- When a publisher is removed, all relays and targets connected to this publisher are disconnected.
- When a relay is removed, all relays and targets connected to this relay are connected to all parents of this relay; that is, either higher level relays or publishers.
- When a target is removed, it is disconnected from all relays and publishers to which it is connected.

Command Syntax:

```
everserve remove peer {-p <peerName>} {-c <communityName>}
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name of the peer you wish to remove from the community. Peer name strings are case sensitive and alphanumeric.
{-c <communityName>}	Y	Name for the community. Community name strings are alphanumeric and can include symbols. Community names are case sensitive.

Example:

The following example illustrates removing the peer "NWoffice" from the community "HR."

```
$ everserve remove peer -p NWoffice -c HR
```

Deleting a Peer

The ***delete peer*** command is used to permanently remove the peer's definition from the system. A peer definition specifies how the peer is known and trusted by the publisher and/or relay. Once a peer's definition is deleted, it is no longer recognized by Everserve and becomes an unknown entity to the community manager, publisher(s), and relay(s). Once deleted, the peer is removed from all communities to which it was associated.

If you wish to withdraw a peer from a community, but want to retain the peer's definition for future inclusion into another community, see [Removing a Peer from a Community](#).

Command Syntax:

```
everserve delete peer {-p <peerName>}
```

Parameters and Options:

Syntax	Required	Description
{-p <peerName>}	Y	Name for the peer. Peer name strings are case sensitive and alphanumeric.

Example:

The following example illustrates deleting the peer "Region2."

```
$ everserve delete peer -p Region2
```

Publishing Commands

The publisher's role is to publish packages to all peers in the community and to monitor return receipts for all packages delivered and logged in the database. Package specifications are created prior to delivery, and are stored in the default directory `\Everserve\server\Packages` directory on the publisher's machine. For a complete guide to creating and publishing packages, refer to the Everserve [Publisher's Guide](#).

Note: By default, package specifications (<packageName.XML>) reside in the `Everserve\server\Packages` directory on the publisher's file system. This is also the default location of the `package_spec.dtd` file used to verify all package elements prior to bundling and sending the package. If package specifications will reside in a directory other than the default, you must copy the `package_spec.dtd` file to the same directory where the package specifications are located.

Delivering Packages

The **deliver** command is used to publish a package to recipient peers.

Delivery Syntax:

```
everserve deliver [immediate]{-f <packageName.xml>}
  [-c <communityName>] [-p <publisherPeerName>] [-i <ID>]
  [-l <peerList>]
```

Parameters and Options:

Syntax	Required	Description
[immediate]	N	Sets package delivery to highest priority. When deliver immediate is used, the package will be opened and executed before all other packages in the target's queue.
{-f <packageName>}	Y	Name of the package you wish to publish to the community. All package file names must include the .XML extension
[-c <communityName>]	N	Name for the community. If more than one community exists for the publisher, you must specify the community name. Community names are case sensitive.

Syntax	Required	Description
[-p <publisherPeerName>]	N	Name of the publisher that has control or responsibility for publishing to the community. If more than one publisher exists in the community, you must specify which publisher is to deliver the package. Peer names are case sensitive.
[-i <ID>]	N	The ID of the package is used to create a familiar name for the delivery, such as "Update_2002". This ID is also used as identifier that accompanies the packages so that the user can distinguish package deliveries. Package IDs may not exceed 50 characters.
[-l <peerList>]	N	By default, the delivery command normally delivers to all targets in the community. The -l switch enables delivery of a package to one or several peers in a given community, as defined by the list of peer names specified. If a relay is specified as the delivery recipient, all peers connected to that relay will receive the delivery. Peer names are case sensitive.

Examples:

The following example illustrates how a publisher sends the package *FormsUpdate.xml* to the community *HR*, by the community publisher, *HRpub*.

```
everserve deliver -f FormsUpdate.xml -c HR -p HRpub
```

The following example illustrates how the publisher *HRpub* sends the package *FormsUpdate* to a list of peers (*peer1*, *peer2*, *peer3*). In this example, an easily recognizable name for the package (and return receipt) is specified as *Update_2002*.

```
everserve deliver -f FormsUpdate.xml -p HRpub -l peer1 peer2 peer3
-i Update_2002
```

Executing Files in Batch Mode

The **run** command is used to execute several Everserve commands from a file. You can create a text file that contains a list of Everserve commands, then run the file as a single command in “batch mode.”

Command Syntax:

```
everserve run {-f <fileName>}
```

Parameters and Options:

Syntax	Required	Description
{-f <fileName>}	Y	Name of the file or package you wish to execute. Filenames are case sensitive (UNIX only).

Example:

The following example illustrates how to create and run a file containing Everserve commands:

1. Create a text file, for example, “HRdeliveries.bat.”
2. List the Everserve commands to include, for example:

```
everserve deliver -f HRorgChart.xml -c HR -p HRpub  
everserve deliver -f HRformsUpdate.xml -c HR -p HRpub  
everserve deliver -f Benefits.xml -c HR -p HRpub
```

3. Save the file.
4. Run the file, for example:

```
everserve run -f HRdeliveries.bat
```

Note: *If the path for the file is not specified, Everserve will look for the file in the \Everserve\server directory.*

The system runs all commands listed in the file, in the order in which they are listed. In the example shown in [Step 2](#), Everserve will initiate three separate package deliveries to the HR community.

Information Services Commands

Everserve provides two sets of commands used to view community and delivery information: “*Show*” commands that allows the use of wildcard characters in the command string, and “*List*” commands. Both *Show* and *List* commands can be used on all Everserve devices, however, the use of wildcard characters is permitted only when using *Show* commands.

Specifying Date Ranges

Everserve lets you specify date ranges when using “*Show*” commands for deliveries, processes, and receipts. The following rules are applicable to date range usage:

- The date parser is locale specific and uses the short date format. In the U.S. locale, the short date format is either mm/dd/yy or mm/dd/yyyy. Leading 0's are permitted, and years may have either 2 or 4 digits.
- The show output shows a date and a time, but the parser only accepts a date and not a time. Everweb provides shorter time ranges in terms of hours, which is not possible with the CLI since it does not parse times.
- To specify a start date for the range, enter “-s <startDate>.” If a start date is not specified, the default is the beginning of the current day.
- To specify an end date, enter “-e <endDate>.” If an end date is not specified, the default is the precise millisecond at which the show command was processed.
- You may specify either startDate and endDate, or startDate or endDate.
- The start date must be less than the end date, regardless of whether the dates are explicit or defaulted.

Show Commands

Show commands are methods of accessing information contained in the local devices' database or file store. Show commands support the use of the wildcard character '*' for those systems that use a database. A wildcard can be used in all names of communities, peers, and identifiers to search for an imprecise name match. If no wildcard character is specified, the system will attempt to find an exact match for the name provided.

Show Communities

Use the **show communities** command to display all communities that the peer is aware of (as with a community manager), or that the peer is a member of (as with a publisher, relay, or target).

Command Syntax

```
everserve show communities [-c <communityName>] [-p <peerName>]
```

Parameters and Options:

Parameter	Required	Description
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.

Example:

The following example illustrates how to show community information for the community “HR.” Sample output for this example is provided:

```
everserve show community -c HR
```

Name	Description	Peers
HR	Human Resources	5

Show Peers

Use the **show peers** command to display all peers that are a member of a community, or to display information about a specific peer.

Command Syntax

```
everserve show peers [-c <communityName>] [-p <peerName>]  
                    [-h <hostName>] [-r <role>]
```

Parameters and Options:

Parameter	Required	Description
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-h <hostname>]	N	The actual location of the machine (peer) on the network. The hostname specified can be a fully qualified hostname, or an IP address. The following are examples of valid hostnames: <ul style="list-style-type: none">• MyMachine• MyMachine.example.com• 012.345.67.89
[-r <role>]	N	Peers may have one role within the context of a community. Peer roles include: <ul style="list-style-type: none">• CM (community manager)• Pub (publisher)• Relay• Target

Examples:

The following example illustrates how to show peer information for the peers in the community "HR." Sample output for these examples is provided:

```
everserve show peers -c HR
```

Peer name	Host name	Community name	Role	Active
HRPub	Colorado_server	HR	Publisher	true
HRcm	Colorado_server	HR	Com.Mgr.	true
Region1	SF_server	HR	Target	true
Region2	boston_server	HR	Target	true
Region3	seattle_server	HR	Target	true

```
5 rows printed.
```

The following example illustrates issuing a wildcard to show all peer names beginning with the letters "Reg.:"

```
everserve show peers -p Reg*
```

Peer Name	Description	Hostname
Region1	Headquarters HR Target	SF_server
Region2	NorthEast HR Target	Boston_server
Region3	NorthWest HR Target	Seattle_server

Show Senders

Use the ***show senders*** command is used to display all publishers or relays in a community from which the relay or target peer receives packages.

Command Syntax

```
everserve show senders [-c <communityName>] [-p <peerName>]
```

Parameters and Options:

Parameter	Required	Description
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.

Examples:

The following example illustrates how to show sender information for all peers in the community “*HR*.”

```
everserve show senders -c HR
```

```
Peer Name Description                      Senders
-----
Region1  Headquarters HR Target          HR_Pub
Region2  NorthEast HR Target          HR_Relay
```

The following example illustrates how to show sender information for the specific peer “*Region1*.”

```
everserve show senders -p Region1
```

```
Name      Description                      Senders
-----
Region1  Headquarters HR Target          HR_Pub
```

The following example illustrates how to show sender information for the specific peer (*Region1*) in the community “*HR*.”

```
everserve show senders -c HR -p Region1
```

Name	Description	Sender
Region1	Headquarters HR Target	HR_Pub

Show Processes

Use the **show processes** command to display all Everserve processes and activities that have run on the peer. When using the **-v** parameter, the system will list each process and all the activities connected to each process.

Command Syntax

```
everserve show processes [-v] [-a][-s <startDate>] [-e <endDate>]  
[-f <find>]
```

Parameters and Options:

Parameter	Required	Description
[-v]	N	Verbose setting to display detailed information about the package delivery.
[-a]	N	Show process records that have been archived that match the criteria.
[-s <startDate>]	N	The beginning date range in which a process was logged. See Specifying Date Ranges for examples of acceptable date-range formats.
[-e <endDate>]	N	The ending date range in which a process was logged. See Specifying Date Ranges for examples of acceptable date-range formats.
[-f <find>]	N	Text string to identify a process. For example, entering the word “Start” would show all “Starting Server” processes.

Example:

The following example shows sample output when executing the show processes command:

```
everserve show processes
```

Description	Start	End	Activities
-----	-----	-----	-----
Starting the server	07/24/02 07:30:57.400	07/24/02 07:30:57.400	1
Starting the server	07/25/02 07:56:25.179	07/25/02 07:56:25.179	1
show communities	07/25/02 09:17:37.354	07/25/02 09:17:37.354	1
deliver -f FormsUpdate.xml	07/25/02 09:18:13.637	07/25/02 09:18:25.494	2
deliver -f MyPackage.xml	07/25/02 09:18:25.564	07/25/02 09:18:25.564	1
show peers -c "HR" -h "S	07/25/02 09:18:25.804	07/25/02 09:18:25.804	1

Show Receipts

Use the **show receipts** command to view information about return receipts.

Command Syntax:

```
everserve show receipts [-a] [-v] [-c <communityName>] [-p <peerName>]
[-i <ID>] [-f <filter>] [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Parameter	Required	Description
[-a]	N	Show those records that match the criteria in the archive tables only.
[-v]	N	Verbose setting to display detailed information about the return receipt.
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-i <ID>]	N	The descriptive text used to identify a package using a short, familiar name, such as "update_2002." Package ID's permit the use of the wildcard character (*). Package IDs may not exceed 50 characters.
[-f <filter>]	N	Permits filtering deliveries based on Pass , Fail , or Pending delivery status.
[-s <startDate>]	N	The beginning date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.
[-e <endDate>]	N	The ending date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.

Examples:

The following example illustrates how to show receipts for the community "HR." Sample output for this example is provided:

```
everserve show receipts -c HR
```

PeerName	Package ID	FileName	Status	Received
Region3	Checking	Everserve Ver	C:\Program Files\Sync	Pass 03/11/02 10:11:45 AM
Region1	Checking	Everserve Ver	C:\Program Files\Sync	Pass 03/11/02 10:12:02 AM
Region2	Checking	Everserve Ver	C:\Program Files\Sync	Pass 03/11/02 10:12:10 AM

The following example illustrates how to show a detailed receipt for a specific peer and a specific delivery. Note that if you do not specify a delivery ID, the system will display all delivery receipts for the peer.

```
everserve show receipts -p Region2 -v -i "Hello Package"
```

```
Peer name: Region2
```

```
Package name: hello package
```

```
Specification file: c:\Program Files\Synchron Netowrks\Everserve\server\Pack-  
ages\hello.xml
```

```
Status: Pass
```

```
Date received: Tue 12 08:29:20 PST
```

```
Specification: c:\hello.txt
```

```
Status: Pass
```

```
Script Return Code: N/A
```

```
stdout:
```

```
stderr:
```

```
Specification: notepad c:\hello.txt
```

```
Status: Pass
```

```
Script Return Code:0
```

```
stdout:
```

```
stderr:
```

Show Deliveries

Use the **show deliveries** command to display all deliveries made to a community or to specific peers, or to view deliveries for a specific package.

Command Syntax

```
everserve show deliveries [-a] [-c <communityName>][-p <peerName>]
                        [-i <ID>] [-f <filter>] [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Parameter	Required	Description
[-a]	N	Show those records that match the criteria in the archive tables only.
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-i <ID>]	N	The descriptive text used to identify a package by a familiar name, such as "update_2002." Package ID's permit the use of the wildcard character (*). Package IDs may not exceed 50 characters.
[-f <filter>]	N	Permits filtering deliveries based on Pass, Fail, or Pending delivery status.
[-s <startDate>]	N	The beginning date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.
[-e <endDate>]	N	The ending date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.

Example:

The following example illustrates show the deliveries made to the community "HR," from 03/09/2002 to 03/11/2002. Sample output for this example is provided:

```
everserve show deliveries -c HR -s 03/09/2002 -e 03/11/2002
```

Description	Date	Stat	Receipts	Targets
deliver -i Checking Everserve Ver	03/10/02 10:15:24 AM	Pass	3	3
deliver -i HR Forms for 2002	03/10/02 10:33:01 AM	Pass	3	3
deliver -i Virus Protection Updat	03/10/02 11:45:13 AM	Pass	3	3
deliver -i Updated Org Chart	03/10/02 12:52:56 PM	Pass	3	3
deliver -i Policy Updates	03/11/02 08:25:05 AM	Pass	3	3

Show Deliverylog

Use the **show deliverylog** command to display all delivery related activities for deliveries made to a community, specific peers, or for a specific package.

Command Syntax

```
everserve show deliverylog [-a] [-c <communityName>][ -p <peerName>]
[-i <ID>] [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Parameter	Required	Description
[-a]	N	Show those records that match the criteria in the archive tables only.
[-c <communityName>]	N	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Either specify the complete community name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric. Either specify the complete peer name, or enter a partial name and use the wildcard character (*) for an imprecise match.
[-i <ID>>]	N	The ID of the package is used to create a short, familiar name for the package, such as "update_2002." Package IDs may not exceed 50 characters.

Parameter	Required	Description
[-s <startDate>]	N	The beginning date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.
[-e <endDate>]	N	The ending date range in which the package was delivered. See Specifying Date Ranges for examples of acceptable date-range formats.

Example:

The following example shows a delivery log for all deliveries made from the publisher's system over a specified period of time.

```
everserve show deliverylog -s 07/28/02 -e 07/29/02
```

Description	Date	Package ID	Peer name	Originator
package delivered	07/29/02 09:21:46	New HR Forms Pa	HRPub	HRPub
package received	07/29/02 09:21:49	New HR Forms Pa	Region1	HRPub
receipt created	07/29/02 09:21:50	New HR Forms Pa	Region1	Region1
package received	07/29/02 09:21:50	New HR Forms Pa	Region3	HRPub
receipt created	07/29/02 09:21:51	New HR Forms Pa	Region3	Region3
package received	07/29/02 09:21:51	New HR Forms Pa	Region2	HRPub
receipt created	07/29/02 09:21:52	New HR Forms Pa	Region2	Region2
receipt received	07/29/02 09:21:52	New HR Forms Pa	HRPub	Region1
receipt received	07/29/02 09:21:53	New HR Forms Pa	HRPub	Region3
receipt received	07/29/02 09:21:53	New HR Forms Pa	HRPub	Region2
package delivered	07/29/02 13:49:26	backup docs	HRPub	HRPub
receipt created	07/29/02 13:49:30	backup docs	Region1	Region1
receipt received	07/29/02 13:49:31	backup docs	HRPub	Region1

List Commands

List commands are used to display miscellaneous information for a given Everserve system.

Listxml

The **listxml** command issued to display detailed information about a community, or for a peer that is a member of a given community.

Command Syntax:

```
everserve listxml {-c <communityName>}[-p <peerName>]
```

Parameters and Options:

Parameter	Required	Description
{-c <communityName>}	Y	Name for the community. Community name strings are case sensitive, alphanumeric, and can include symbols. Wildcards are not allowed or recognized.
[-p <peerName>]	N	Name for the peer. Peer name strings are case sensitive and alphanumeric.

List Roles

The **list roles** command is used to display all assignable roles supported by Everserve.

Command Syntax:

```
everserve list roles
```

Listing Environment Settings

The **env** command is used to display Everserve's environment variables.

Command Syntax:

```
everserve env
```

Listing Everserve Version

The **version** command is used to display which version of Everserve is currently installed and running on the device.

Command Syntax:

```
everserve version
```

Help

The **help** command is used to display an alphabetical listing of all Everserve commands and their parameters.

Command Syntax:

```
everserve help
```

Creating a Community

The System Administrator is responsible for planning the community topology before installing Everserve on any device other than the community manager system, and is also responsible for creating and maintaining community definitions.

Once the topology has been defined and mapped-out, Everserve can then be installed on all devices that will be a member of a community. At install time, role capabilities are chosen for each device based on the role the device will have in the community. Therefore, it is important to ensure all devices receive the correct role capabilities at install time. For information on how to plan an installation for a community, see [Planning Your Everserve Community](#) in this guide.

This section outlines the process of creating a complex community topology that includes redundant routing, multiple levels of relays, and multiple community managers and publishers. This section includes step-by-step instructions on how to create this “Example” community using Everserve’s command line interface, and how to deliver various package types to the entire community, and how to deliver packages to individual peers in the community.

This topics in this section include:

- [Community Hierarchy](#)
- [Creating the “Example” Community](#)
- [Configuring Backup Publishers](#)
- [Publishing Packages](#)

Community Hierarchy

The example community depicted in [Figure 4](#) is configured with two community managers, two publishers, two tiers of relays, several targets, and redundant routing. The following sections describe how to create the “Example” community illustrated below:

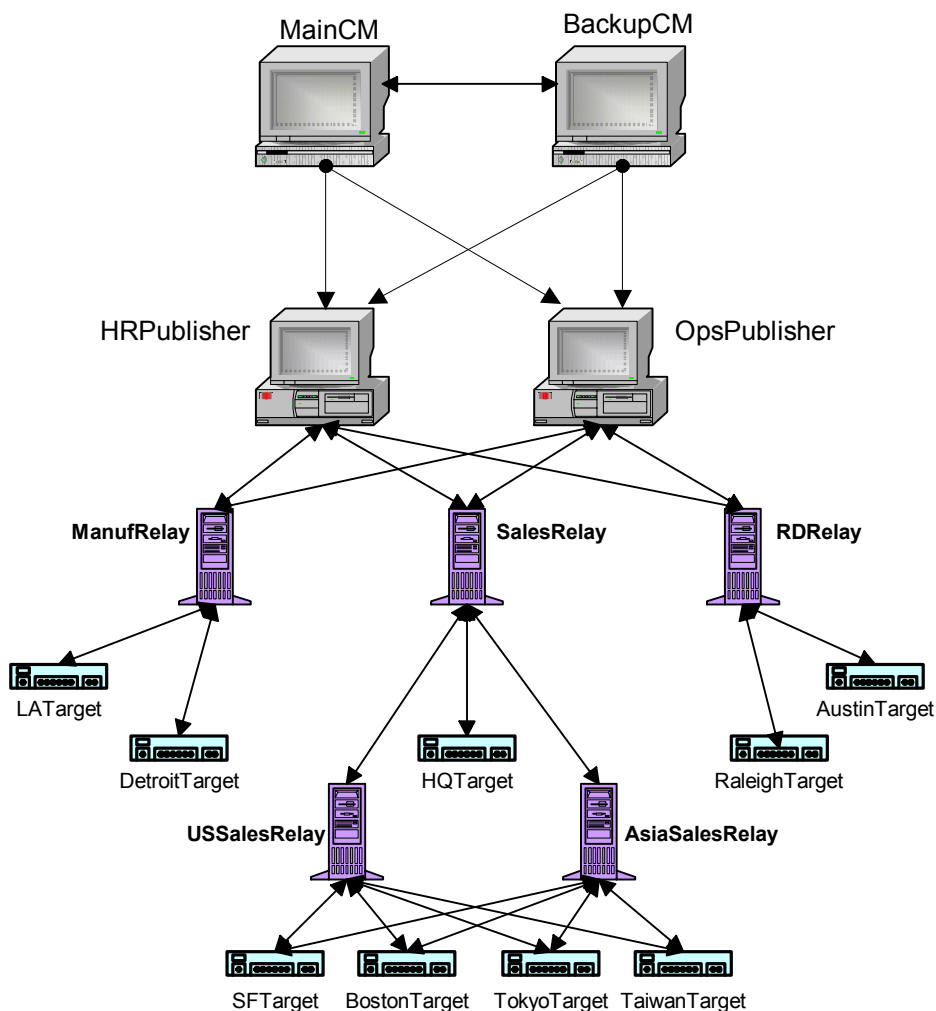


Figure 4 Hierarchy of Example Community

Multiple Community Managers

The example community shown in [Figure 4](#) is configured with two community managers to allow for a main community manager and a backup community manager. The main community manager is used for all community operations, and is used to create the backup community manager. Without a backup community manager, any future maintenance of the community by the main community manager becomes impossible should the main community manager experience a catastrophic failure. In which case, the backup community manager has full access to the community definition and can perform all maintenance tasks as needed.

Multiple Publishers

The community shown in [Figure 4](#) uses two publishers to demonstrate how two separate System Administrators could publish different types of packages to an entire community. These publishers can also be used in a separate community as a way to provide as backup to the other in the case of a catastrophic failure on either of the publishers.

For this example, the HR Publisher will publish documents related to employees, such as reports and forms. The Operations Publisher will publish packages for maintenance of systems, such as applications and patches.

Tiered Relays and Multiple Targets

Multiple relays allow faster distribution to a large number of targets. For the purpose of brevity in this example, only a small number of targets will be configured. However, it is highly probable that many more targets would exist in a real world scenario. As shown in [Figure 4](#), the relays are designated as follows:

- The Manufacturing Relay (Manuf) will distribute packages to all sites in the manufacturing organization.
- The Sales Relay will be used to distribute packages to all sales sites, which also includes a second tier of relays. This second tier of relays will include redundant routes; using two relays to go to the same set of targets to ensure packages are received in a timely manner.
- The R&D Relay will distribute packages to all sites in the R&D organization.

Creating the “Example” Community

This section includes the exact steps to create the community “Example” shown in [Figure 4](#). The very first step is to create an initial community, and the community’s main and backup community managers. The initial community will contain all of the members described in the preceding sections. Hostnames are shown as descriptions, not real hostnames. A sample text file containing all of the commands listed in this section is provided in [Appendix A: Sample Script for Creating Communities](#).

Using the Everserve Command Shell

Everserve provides a shell enabling command execution without having to prefix each command with “everserve.” All of the commands in this section can be run outside of the Everserve shell by placing the “everserve” command in front of the command, however, running within the shell will improve system performance.

To Enter the Everserve Command Shell:

1. From the command prompt, execute the `everserve` command to invoke the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All  
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

Create the Community

The following sections list the exact steps used to create the “Example” community. This community will consist of two community managers, two publishers, five relays, and several targets.

Create the Community and Community Managers

The following instructions describe how to create the community named “Example,” and the two community managers; “MainCM” and “BackupCM.”

1. Create the community (Example) and main community manager (MainCM). Provide a description of the community for identification purposes:

```
create community -c Example -p MainCM -d "Example Community"
```


The system displays the following:

```
Created community "Example" - "Example Community".
Created local peer name = "MainCM" description = "Default community
manager for example" networkAddress = "Everserve@MainCMhostname"
Added peer "MainCM" to community "Example" with the role of community
manager.
Created Everserve seed file "Example.zip" in directory "C:/Program
Files/Everserve/Server\".
```

2. Create the peer that will be the backup community manager (BackupCM) and provide a description of the peer.

```
create peer -p BackupCM -h BackupCMhostname -d "Backup community
manager"
```

The system displays the following:

```
Created peer name = "BackupCM" description = "Backup Community Manager"
networkAddress = "backupcmhostname"
```

3. Add the BackupCM peer to the Example community with the role of community manager.

```
add peer -p BackupCM -c Example -r CM
```

The system displays the following:

```
Added peer "BackupCM" to community "Example" with the role of community
manager.
```

Create the Publishers

The following instructions describe how to create the community publishers; "HRPublisher" and "OpsPublisher."

1. Create the peer that will be the HR publisher (HRPublisher).

```
create peer -p HRPublisher -h HRPublisherhostname -d "HR Publisher"
```

The system displays the following:

```
Created peer name = "HRPublisher" description = "HR Publisher"
networkAddress = "hrpublisherhostname"
```

2. Add the HRPublisher peer to the Example community with the role of publisher.

```
add peer -p HRPublisher -c Example -r Publisher
```

The system displays the following:

```
Added peer "HRPublisher" to community "Example" with the role of  
publisher.
```

3. Create the peer that will be the operations publisher (OpsPublisher).

```
create peer -p OpsPublisher -h OpsPublisherhostname -d "Operations  
Publisher"
```

The system displays the following:

```
Created peer name = "OpsPublisher" description = "Operations Publisher"  
networkAddress = "opspublisherhostname"
```

4. Add the OpsPublisher peer to the Example community with the role of publisher.

```
add peer -p OpsPublisher -c Example -r Publisher
```

The system displays the following:

```
Added peer "OpsPublisher" to community "Example" with the role of  
publisher.
```

Create the Relays

The following instructions describe how to create the first tier of community relays; "SalesRelay," "ManufRelay," and RDRelay," and the second tier of relays; "USSalesRelay" and AsiaSalesRelay."

Create the First Tier of Relays:

1. Create the peer that will be the sales relay (SalesRelay).

```
create peer -p SalesRelay -h SalesRelayhostname -d "Sales Main Relay"
```

The system displays the following:

```
Created peer name = "SalesRelay" description = "Sales Main Relay"  
networkAddress = "salesrelayhostname"
```

2. Add the SalesRelay peer to the Example community with the role of relay, with the senders of HRPublisher and OpsPublisher.

```
add peer -p SalesRelay -c Example -r Relay -s HRPublisher OpsPublisher
```

The system displays the following:

```
Added peer "SalesRelay" to community "Example" with the role of relay.
```

3. Create the peer that will be the manufacturing relay (ManufRelay).

```
create peer -p ManufRelay -h ManufRelayhostname -d "Manufacturing Main  
Relay"
```

The system displays the following:

```
Created peer name = "ManufRelay" description = "Manufacturing Main  
Relay" networkAddress = "manufrelayhostname"
```

4. Add the ManufRelay peer to the Example community with the role of relay, with the senders of HRPublisher and OpsPublisher.

```
add peer -p ManufRelay -c Example -r Relay -s HRPublisher OpsPublisher
```

The system displays the following:

```
Added peer "ManufRelay" to community "Example" with the role of relay.
```

5. Create the peer that will be the R&D relay (RDRelay).

```
create peer -p RDRelay -h RDRelayhostname -d "R&D Main Relay"
```

The system displays the following:

```
Created peer name = "RDRelay" description = "R&D Main Relay"  
networkAddress = "rdrelayhostname"
```

6. Add the RDRelay peer to the Example community with the role of relay, with the senders of HRPublisher and OpsPublisher.

```
add peer -p RDRelay -c Example -r Relay -s HRPublisher OpsPublisher
```

The system displays the following:

```
Added peer "RDRelay" to community "Example" with the role of relay.
```

Create the Second Tier of Relays:

Note: The following steps will create, add, and connect the second tier relays to the first tier SalesRelay.

1. Create the peer that will be the US sales relay (USSalesRelay).

```
create peer -p USSalesRelay -h USSalesRelayhostname -d "US Sales Relay"
```

The system displays the following:

```
Created peer name = "USSalesRelay" description = "US Sales Relay"
networkAddress = "ussalesrelayhostname"
```

2. Add the USSalesRelay peer to the Example community with the role of Relay, with the sender of SalesRelay.

```
add peer -p USSalesRelay -c Example -r Relay -s SalesRelay
```

The system displays the following:

```
Added peer "USSalesRelay" to community "Example" with the role of relay.
```

3. Create the peer that will be the Asia sales relay (AsiaSalesRelay).

```
create peer -p AsiaSalesRelay -h AsiaSalesRelayhostname -d "Asia Sales
Relay"
```

The system displays the following:

```
Created peer name = "AsiaSalesRelay" description = "Asia Sales Relay"
networkAddress = "asiasalesrelayhostname"
```

4. Add the AsiaSalesRelay peer to the Example community with the role of relay, with the sender of SalesRelay.

```
add peer -p AsiaSalesRelay -c Example -r Relay -s SalesRelay
```

The system displays the following:

```
Added peer "AsiaSalesRelay" to community "Example" with the role of
relay.
```

5. Verify that all five relays show the correct senders by using the show senders command.

```
show senders -c Example
```

The system displays the following:

Peer name	Community name	Senders
AsiaSalesRelay	Example	SalesRelay (relay)
ManufRelay	Example	HRPublisher(pub), OpsPublisher(pub)
RDRelay	Example	HRPublisher(pub), OpsPublisher(pub)
SalesRelay	Example	HRPublisher(pub), OpsPublisher(pub)
USSalesRelay	Example	SalesRelay(relay)

5 rows printed.

Create the Targets

Note: If a role is not specified for the device being added to a community, the device will be assigned the default role of target. Therefore, when adding targets to a community, the -r parameter is optional.

1. Create the peer that will be the head quarters target (HQTarget).

```
create peer -p HQTarget -h HQTargethostname -d "HQ Target"
```

The system displays the following:

```
Created peer name = "HQTarget" description = "HQ Target" networkAddress =
"hqtargethostname"
```

2. Add the HQTarget peer to the Example community, with a role of target, and its sender as the SalesRelay.

```
add peer -p HQTarget -c Example -s SalesRelay
```

The system displays the following:

```
Added peer "HQTarget" to community "Example" with the role of target.
```

3. Create the peer that will be the San Francisco target (SFTarget) for the sales sub-relays.

```
create peer -p SFTarget -h SFTargethostname -d "SF Sales Office Target"
```

The system displays the following:

```
Created peer name = "SFTarget" description = "SF Sales Office Target"
networkAddress = "sftargethostname"
```

4. Add the SFTarget peer to the Example community, with a role of target and its senders as the USSalesRelay and the AsiaSalesRelay. This will create a redundant route to SFTarget whenever a package is sent to the community or to SalesRelay specifically. This will ensure that all sales targets will receive their packages if the Internet connection between one of the sub-relays goes down.

```
add peer -p SFTarget -c Example -s USSalesRelay AsiaSalesRelay
```

The system displays the following:

```
Added peer "SFTarget" to community "Example" with the role of target.
```

5. Create the peer that will be the Boston target (BostonTarget) for the sales sub-relays.

```
create peer -p BostonTarget -h BostonTargethostname -d "Boston Sales
Office Target"
```

The system displays the following:

```
Created peer name = "BostonTarget" description = "Boston Sales Office
Target" networkAddress = "bostontargethostname"
```

6. Add the BostonTarget peer to the Example community, with a role of target and its senders as USSalesRelay and the AsiaSalesRelay.

```
add peer -p BostonTarget -c Example -s USSalesRelay AsiaSalesRelay
```

The system displays the following:

```
Added peer "BostonTarget" to community "Example" with the role of target.
```

7. Create the peer that will be the Taiwan target (TaiwanTarget) for the sales sub-relays.

```
create peer -p TaiwanTarget -h TaiwanTargethostname -d "Taiwan Sales  
Office Target"
```

The system displays the following:

```
Created peer name = "TaiwanTarget" description = "Taiwan Sales Office  
Target" networkAddress = "taiwantargethostname"
```

8. Add the TaiwanTarget peer to the Example community, with a role of target and its senders as USSalesRelay and the AsiaSalesRelay.

```
add peer -p TaiwanTarget -c Example -s USSalesRelay AsiaSalesRelay
```

The system displays the following:

```
Added peer "TaiwanTarget" to community "Example" with the role of target.
```

9. Create the peer that will be the Tokyo target (TokyoTarget) for the sales sub-relays.

```
create peer -p TokyoTarget -h TokyoTargethostname -d "Tokyo Sales Office  
Target"
```

The system displays the following:

```
Created peer name = "TokyoTarget" description = "Tokyo Sales Office  
Target" networkAddress = "tokyotargethostname"
```

10. Add the TokyoTarget peer to the Example community, with a role of target and its senders as USSalesRelay and the AsiaSalesRelay.

```
add peer -p TokyoTarget -c Example -s USSalesRelay AsiaSalesRelay
```

The system displays the following:

```
Added peer "TokyoTarget" to community "Example" with the role of target.
```

11. Create the peer that will be the Los Angeles target (LATarget) for the manufacturing relay.

```
create peer -p LATarget -h LATargethostname -d "LA Manufacturing Target"
```

The system displays the following:

```
Created peer name = "LATarget" description = "LA Manufacturing Target"  
networkAddress = "latargethostname"
```

12. Add the LATarget peer to the Example community, with a role of target and its sender as the ManufRelay.

```
add peer -p LATarget -c Example -s ManufRelay
```

The system displays the following:

```
Added peer "LATarget" to community "Example" with the role of target.
```

13. Create the peer that will be the Detroit target (DetroitTarget) for the manufacturing relay.

```
create peer -p DetroitTarget -h DetroitTargethostname -d "Detroit  
Manufacturing Target"
```

The system displays the following:

```
Created peer name = "DetroitTarget" description = "Detroit Manufacturing  
Target" networkAddress = "detroittargethostname"
```

14. Add the DetroitTarget peer to the Example community, with a role of target and its sender as the ManufRelay.

```
add peer -p DetroitTarget -c Example -s ManufRelay
```

The system displays the following:

```
Added peer "DetroitTarget" to community "Example" with the role of  
target.
```

15. Create the peer that will be the Raleigh target (RaleighTarget) for the R&D relay.

```
create peer -p RaleighTarget -h RaleighTargethostname -d "Raleigh R&D  
Office Target"
```

The system displays the following:

```
Created peer name = "RaleighTarget" description = "Raleigh R&D Office  
Target" networkAddress = "raleightargethostname"
```


16. Add the RaleighTarget peer to the Example community, with its sender as the RDRelay.

```
add peer -p RaleighTarget -c Example -s RDRelay
```

The system displays the following:

```
Added peer "RaleighTarget" to community "Example" with the role of
target.
```

17. Create the peer that will be the Austin target (AustinTarget) for the R&D relay.

```
create peer -p AustinTarget -h Austintargethostname -d "Austin R&D
Office Target"
```

The system displays the following:

```
Created peer name = "AustinTarget" description = "Austin R&D Office
Target" networkAddress = "austintargethostname\"
```

18. Add the AustinTarget peer to the Example community, with its sender as the RDRelay.

```
add peer -p AustinTarget -c Example -s RDRelay
```

The system displays the following:

```
Added peer "AustinTarget" to community "Example" with the role of target.
```

19. View the community configuration using the show peers command.

```
show peers -c Example
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
-----	-----	-----	-----	-----
AsiaSalesRelay	asiasalesrelayhostname	Example	Relay	false
AustinTarget	austintargethostname	Example	Target	false
BackupCM	backupcmhostname	Example	Com. Mgr.	false
BostonTarget	bostontargethostname	Example	Target	false
DetroitTarget	detroittargethostname	Example	Target	false
HQTarget	hqtargethostname	Example	Target	false
HRPublisher	hrpublisherhostname	Example	Publisher	false
LATarget	latargethostname	Example	Target	false

MainCM	maincmhostname	Example	Com. Mgr.	true
ManufRelay	manufrelayhostname	Example	Relay	false
OpsPublisher	opspublisherhostname	Example	Publisher	false
RDRelay	rdrelayhostname	Example	Relay	false
RaleighTarget	raleightargethostname	Example	Target	false
SFTarget	sftargethostname	Example	Target	false
SalesRelay	salesrelayhostname	Example	Relay	false
TaiwanTarget	taiwantargethostname	Example	Target	false
TokyoTarget	tokyotargethostname	Example	Target	false
USSalesRelay	ussalesrelayhostname	Example	Relay	false

18 rows printed.

Note: The Active flag for each peer will be false until the peer joins the community. The exception to this are peers that are located on the same system as the community definition (in this example, MainCM is automatically activated).

20. Exit the Everserve shell.

```
exit
```

Joining the Community

Before a peer can actively participate in the community, a challenge-response joining operation must take place. A seed file was created on the peer that serves as the Main CM for the Example community and lives in that Everserve\Server directory. This seed file will set up a private/public key pair between the community manager(s) and each peer in the community, ensuring secure transmission of packages. See [Activating a New Everserve Device](#) in this guide for additional information.

Note: A peer is not considered an active member of a community until the join process is complete.

The following steps must be performed on all devices in the community.

1. Copy the seed file for the community, example.zip, to the Everserve\server directory of each peer in your community.

- On each peer in the Example community, issue the `join` command.

```
everserve join -c Example
```

The system displays the following:

```
Trying to join a new community, please wait.
Successfully joined community "Example".
```

- Once all peers have successfully joined the Example community, verify their Active status by using the `show peers` command on the community manager peer.

```
everserve show peers -c Example
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
AsiaSalesRelay	asiasalesrelayhostname	Example	Relay	true
AustinTarget	austintargethostname	Example	Target	true
BackupCM	backupcmhostname	Example	Com. Mgr.	true
BostonTarget	bostontargethostname	Example	Target	true
DetroitTarget	detroittargethostname	Example	Target	true
HQTarget	hqtargethostname	Example	Target	true
HRPublisher	hrpublisherhostname	Example	Publisher	true
LATarget	latargethostname	Example	Target	true
MainCM	maincmhostname	Example	Com. Mgr.	true
ManufRelay	manufrelayhostname	Example	Relay	true
OpsPublisher	opspublisherhostname	Example	Publisher	true
RDRelay	rdrelayhostname	Example	Relay	true
RaleighTarget	raleightargethostname	Example	Target	true
SFTarget	sftargethostname	Example	Target	true
SalesRelay	salesrelayhostname	Example	Relay	true
TaiwanTarget	taiwantargethostname	Example	Target	true
TokyoTarget	tokyotargethostname	Example	Target	true
USSalesRelay	ussalesrelayhostname	Example	Relay	true

18 rows printed.

The example community definition is now complete with all peers in the community as active members. The publishers for this community can now begin to publish packages to the entire community (see [Publishing to an Entire Community](#)), to a relay in the community (see [Publishing to a Relay](#)), to a single target (see [Publishing to a Single Target](#)), or to a list of targets (see [Publishing to a Lists of Targets](#)).

Configuring Backup Publishers

Similar to having backup community managers for a community, you may wish to have the ability for one or more publishers to serve as a backup for the community's main publisher. This requires all package specifications and datasets used by the main publisher for a community to be available on other publishers that will serve as a backup should the main publisher experience a catastrophic failure. The easiest method to synchronize backup publishers with the main publisher is to create a small community that contains the backup publishers and use Everserve to deliver packages from the main publisher to this community on a regular basis.

Everserve supports multiple peers on a single device as long as each peer on the device has a unique peer name. This allows you to create a new target peer on the devices that will be used as backup publishers, then publish packages to the target peers on the backup publishers. Since all peers must be a part of, and have a role in a community to receive packages, you would create a new community consisting of the community managers, the main publisher, and the newly created target peers on each backup publisher. Because this is a new community, each peer in the community would need to join the community to become a trusted member and receive packages. This configuration allows the main publisher to continue to publish packages to the primary community (such as the community "Example"), and also publish a different set of packages to the smaller community of backup publishers.

This method of creating a backup publisher configuration allows the main publisher to deliver "special packages" to the backup publishers only. A "special package" would be files and data that you may not want to distribute to all targets in a community, but are needed by publishers to publish packages to a community. This method ensures that all package specifications and datasets used by the main publisher is delivered to each publisher that will be configured as a backup.

In the "Example" community, there are two publishers that serve different roles in the community; one that publishes HR packages and one that publishes operations packages. As shown in [Figure 5](#), the HRPublisher Backup community that consists of the existing MainCM, BackupCM, and HRPublisher, and a new HRBackup target. The HRBackup target will be a newly created peer that resides on the same device as OpsPublisher.

[Figure 6](#) illustrates the OpsPublisher Backup community that consists of the existing MainCM, BackupCM, and OpsPublisher, and a new OpsBackup target. The OpsBackup target will be a newly created peer that resides on the same device as HRPublisher.

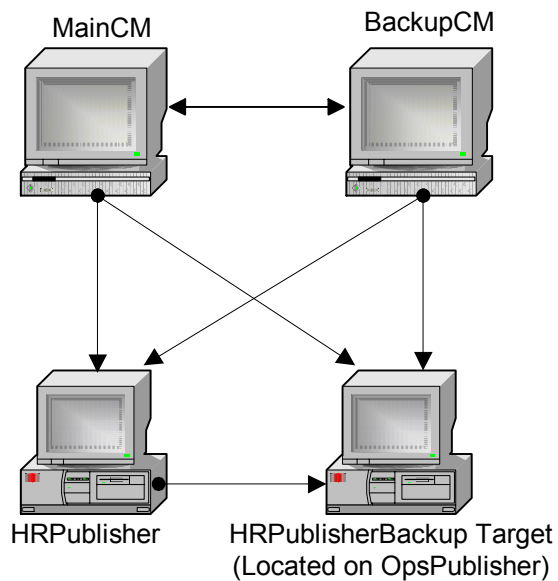


Figure 5 HRPublisher Backup Community

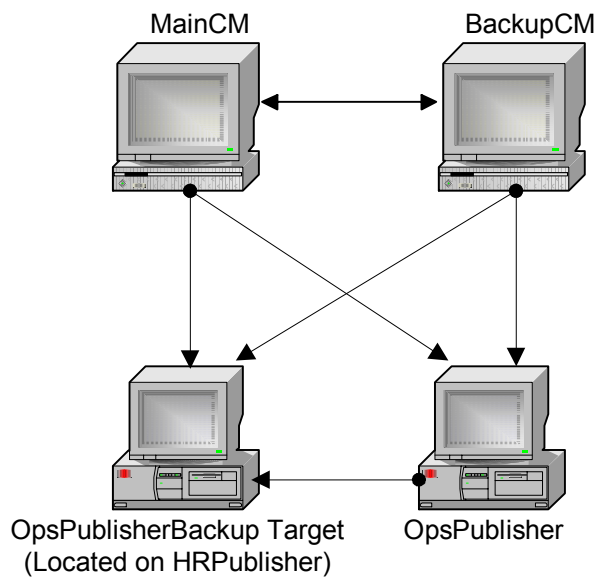


Figure 6 OpsBackup Community Configuration

Creating Backup Communities

The following instructions describe how to create two backup publisher communities; one for HRPublisher, and one for OpsPublisher.

Note: Perform the following steps from the main community manager (MainCM) device to create the communities to be used as backup targets for each publisher.

1. Enter the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All  
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

2. Create a community that will serve as the backup HR Publisher mechanism (HRPublisherBackup), using the MainCM community manager peer as the community manager for this community.

```
create community -c HRPubBackup -p MainCM -d "Community for HRPublisher  
Backup"
```

The system displays the following:

```
Created community "HRPubBackup" - "Community for HRPublisher Backup".
```

```
Added peer "MainCM" to community "HRPubBackup" with the role of community  
manager.
```

```
Created Everserve seed file "HRPubBackup.zip" in directory "C:/  
Everserve/Server\".
```

3. Create a community that will serve as the backup Operations Publisher mechanism (OpsPublisherBackup), using the MainCM community manager peer as the community manager for this community.

```
create community -c OpsPubBackup -p MainCM -d "Community for  
OpsPublisher Backup"
```

The system displays the following:

```
Created community "OpsPubBackup" - "Community for OpsPublisher Backup".
Added peer "MainCM" to community "OpsPubBackup" with the role of
  community manager.
Created Everserve seed file "OpsPubBackup.zip" in directory "C:/
  Everserve/Server\".
```

4. View the three communities using the show communities command.

```
show communities
```

The system displays the following:

Community name	Description	Peers
Example	Example Community	18
HRPubBackup	Community for HRPublisher Backup	4
OpsPubBackup	Community for OpsPublisher Backup	4

3 rows printed.

5. Add the BackupCM to the HRPubBackup community with the role of community manager.

```
add peer -p BackupCM -c HRPubBackup -r CM
```

The system displays the following:

```
Added peer "BackupCM" to community "HRPubBackup" with the role of
  community manager.
```

6. Add the BackupCM to the OpsPubBackup community with the role of community manager.

```
add peer -p BackupCM -c OpsPubBackup -r CM
```

The system displays the following:

```
Added peer "BackupCM" to community "OpsPubBackup" with the role of
  community manager.
```

7. Add HRPublisher to the HRPubBackup community with the role of publisher.

```
add peer -p HRPublisher -c HRPubBackup -r Publisher
```

The system displays the following:

```
Added peer "HRPublisher" to community "HRPubBackup" with the role of  
publisher.
```

8. Add OpsPublisher to the OpsPubBackup community with the role of publisher.

```
add peer -p OpsPublisher -c OpsPubBackup -r Publisher
```

The system displays the following:

```
Added peer "OpsPublisher" to community "OpsPubBackup" with the role of  
publisher.
```

9. Create a target peer on the OpsPublisher device to serve as the storage mechanism for packages published by HRPublisher (the packages are used to synchronize the publishers with each other if a backup publisher is needed).

```
create peer -p HRBackupTarget -h OpsPublisherhostname -d "Backup Target  
for HR Publisher on OpsPublisher"
```

The system displays the following:

```
Created peer name = "HRBackupTarget" description = "Backup Target for HR  
Publisher on OpsPublisher" networkAddress = "opspublisherhostname"
```

10. Create a target peer to serve as the storage mechanism for the OpsPublisher to be used if a backup is needed.

```
create peer -p OpsBackupTarget -h HRPublisherhostname -d "Backup Target  
for OpsPublisher on HR Publisher"
```

The system displays the following:

```
Created peer name = "OpsBackupTarget" description = "Backup Target for  
OpsPublisher on HR Publisher" networkAddress = "hrpublisherhostname"
```


11. Add the HRBackupTarget peer to the HRPubBackup community as a target. Note that the sender option does not have to be specified, as there is only one sender in this community – the publisher.

```
add peer -p HRBackupTarget -c HRPubBackup
```

The system displays the following:

```
Added peer "HRBackupTarget" to community "HRPubBackup" with the role of
target.
```

12. Add the OpsBackupTarget peer to the OpsPubBackup community as a target.

```
add peer -p OpsBackupTarget -c OpsPubBackup
```

The system displays the following:

```
Added peer "OpsBackupTarget" to community "OpsPubBackup" with the role
of target.
```

13. View all peers in each of the communities by using the show peers command.

```
show peers -c HRPubBackup
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
-----	-----	-----	-----	-----
BackupCM	backupcmhostname	HRPubBackup	Com. Mgr.	false
HRBackupTarget	opspublisherhostname	HRPubBackup	Target	false
HRPublisher	hrpublisherhostname	HRPubBackup	Publisher	false
MainCM	maincmhostname	HRPubBackup	Com. Mgr.	true

4 rows printed.

```
show peers -c OpsPubBackup
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
BackupCM	backupcmhostname	OpsPubBackup	Com. Mgr.	false
MainCM	maincmhostname	OpsPubBackup	Com. Mgr.	true
OpsBackupTarget	hrpublisherhostname	OpsPubBackup	Target	false
OpsPublisher	opspublisherhostname	OpsPubBackup	Publisher	false

4 rows printed.

You now have two small communities that can be used to keep the packages and their required data synchronized between the two publishers by publishing packages to each other on a regular basis. As with the Example community, you will need to have each peer join their respective communities before they can become active in the community.

14. Copy the HRPubBackup.zip and OpsPubBackup.zip seed files from the MainCM Everserve\server directory to the Everserve\server directory on the following hosts:

- backupcmhostname
- hrpublisherhostname
- opspublisherhostname

15. Join both communities on each of the systems named above.

```
everserve join -c HRPubBackup
```

The system displays the following:

```
Trying to join a new community, please wait.
Successfully joined community "HRPubBackup".
```

```
everserve join -c OpsPubBackup
```

The system displays the following:

```
Trying to join a new community, please wait.
Successfully joined community "OpsPubBackup".
```

16. Verify that all peers have successfully joined the two communities by using the `show peers` command on the MainCM device.

```
show peers -c HRPubBackup
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
BackupCM	backupcmhostname	HRPubBackup	Com. Mgr.	true
HRBackupTarget	opspublisherhostname	HRPubBackup	Target	true
HRPublisher	hrpublisherhostname	HRPubBackup	Publisher	true
MainCM	maincmhostname	HRPubBackup	Com. Mgr.	true

4 rows printed.

```
show peers -c OpsPubBackup
```

The system displays the following:

Peer name	Host name	Community name	Role	Active
BackupCM	backupcmhostname	OpsPubBackup	Com. Mgr.	true
MainCM	maincmhostname	OpsPubBackup	Com. Mgr.	true
OpsBackupTarget	hrpublisherhostname	OpsPubBackup	Target	true
OpsPublisher	opspublisherhostname	OpsPubBackup	Publisher	true

4 rows printed.

17. Exit the Everserve shell.

```
exit
```

Publishing Packages

Once the community peers have joined the community, packages can now be sent to any or all peers defined. Packages sent can be sent to an entire community, a subset of peers, or to a single peer.

Publishing to an Entire Community

Everserve is typically used to perform some action on each target in a community. You may wish to update a file, get the OS version of each system, view the version of Everserve running on each system, or install an application.

The following instructions will publish a package that will return the OS version of each target in the community.

1. Create a package called GetWinOS.xml that will retrieve the Windows version of the OS. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE spec-container SYSTEM "package_spec.dtd">
<spec-container name="Retrieve Version of Windows OS" version="false"
  delta="false">
  <spec-version version="2.0" earliest="1.0"/>
  <command-spec commandline="ver" success-codes="0"/>
</spec-container>
```

2. Enter the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

- Deliver the GetWinOS package to the Example community from the OpsPublisher system.

```
deliver -f GetWinOS.xml -c Example -p OpsPublisher -i "Getting Windows OS
Version"
```

The system displays the following:

```
Package "...\\Everserve\\Server\\Packages\\GetWinOS.xml" with ID "Getting
Windows OS Version" queued for delivery to "Example" by
"OpsPublisher".
```

- Review the status of the delivery using show deliveries and show receipts commands.

```
show deliveries
```

The system displays the following:

Description	Date	Status	Receipts	Targets
deliver -i "Getting Windows OS Ver	07/08/02 17:40:44	Fail	9	9

1 row printed.

```
show receipts -i "Getting Windows OS Version"
```

The system displays the following:

Peer name	Package ID	File name	Status	Received
TaiwanTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:41:01
SFTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:41:01
BostonTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:41:01
TokyoTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:41:01
LATarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:40:59
DetroitTarget	Getting Windows	GetWinOS.xml	Fail	07/08/02 17:40:59
HQTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:40:57
RaleighTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:40:57
AustinTarget	Getting Windows	GetWinOS.xml	Pass	07/08/02 17:40:56

9 receipts printed.

5. View the contents of a return receipt for a target that passed.

```
show receipts -v -i "Getting Windows OS Version" -p TaiwanTarget
```

The system displays the following:

```
Peer name: TaiwanTarget
Package name: Getting Windows OS Version
Specification file: ...\Everserve\Server\Packages\GetWinOS.xml
Status: Pass
Date received: Mon Jul 08 17:41:01 PDT 2002
Specification: ver
Status: Pass
Script Return Code: 0
stdout:
Microsoft Windows 2000 [Version 5.00.2195]
stderr:
```

6. View the contents of a return receipt for a target that failed.

```
show receipts -v -i "Getting Windows OS Version" -p DetroitTarget
```

The system displays the following:

```
Peer name: DetroitTarget
Package name: Getting Windows OS Version
Specification file: ...\Everserve\Server\Packages\GetWinOS.xml
Status: Fail
Date received: Mon Jul 08 17:40:59 PDT 2002
Specification: ver
Status: Fail
Script Return Code: 1
stdout:
stderr: 'ver' is not recognized as an internal or external command,
operable program or batch file.
```

Publishing to a Relay

The following instructions illustrates how to publish a package to the Main Sales Relay only. The Main Sales Relay will then pass on the report to all peers connected to the relay, which then delivers the sales report to all sales offices. Note that this package specification includes the use of a system variable.

1. Create the JuneRpt.xml package specification. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE spec-container SYSTEM "package_spec.dtd">
<spec-container name="June Sales Report" version="true" delta="true">
  <spec-version version="2.0" earliest="1.0"/>
  <file-spec source="c:\Reports\JuneSales.RPT"
    target="{${SystemRoot}}\Reports\JunesSales.RPT"/>
</spec-container>
```

2. Enter the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

3. Deliver the package to the sales offices only using the HRPublisher system.

```
deliver -f JuneRPT.xml -c Example -l SalesRelay -p HRPublisher
```

The system displays the following:

```
Package "C:\Everserve\Server\Packages\JuneRPT.xml" with ID "Package
JuneRPT.xml" queued for delivery to
```

```
"Example" by "HRPublisher".
```

4. Review the status of the delivery.

```
show deliveries -i "Package JuneRPT.xml"
```

The system displays the following:

Description	Date	Status	Receipts	Targets
"Package JuneRPT.xml"	07/08/02 18:02:28	Pass	5	5

1 row printed.

5. View a list of return receipts for a delivery:

```
everserve show receipts -i "Package JuneRPT.xml"
```

The system displays the following:

Peer name	Package ID	File name	Status	Received
BostonTarget	Package JuneRPT	JuneRPT.xml	Pass	07/08/02 18:02:38
SFTarget	Package JuneRPT	JuneRPT.xml	Pass	07/08/02 18:02:38
TaiwanTarget	Package JuneRPT	JuneRPT.xml	Pass	07/08/02 18:02:38
TokyoTarget	Package JuneRPT	JuneRPT.xml	Pass	07/08/02 18:02:38
HQTarget	Package JuneRPT	JuneRPT.xml	Pass	07/08/02 18:02:35

5 receipts printed.

6. View the contents of a specific return receipt.

```
show receipts -v -i "Package JuneRPT.xml" -p SFTarget
```

The system displays the following:

```
Peer name: SFTarget
Package name: Package JuneRPT.xml
Specification file: ...\Everserve\Server\Packages\JuneRPT.xml
Status: Pass
Date received: Mon Jul 08 18:02:38 PDT 2002
Specification: {$SystemRoot}\Reports\JunesSales.RPT
Status: Pass
Script Return Code: N/A
stdout:
stderr:
```


Publishing to a Single Target

Everserve supports package delivery to a single target rather than the entire community using the `-l` parameter. The following steps illustrate creating the package "RevenueProjections.xml" and delivering this package to the HQTarget.

1. Create a package called RevenueProjections.xml that will send the latest revenue forecast report to recipient peers. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE spec-container SYSTEM "package_spec.dtd">
<spec-container name="June Sales Report" version="true" delta="true">
  <spec-version version="2.0" earliest="1.0"/>
  <file-spec source="c:\Reports\RevenueProjections.RPT"
    target="{${SystemRoot}}\Reports\RevenueProjections.RPT"/>
</spec-container>
```

2. Enter the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All
  Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

3. Deliver the RevenueProjections.xml package to the HQTarget from the OpsPublisher system.

```
deliver -f RevenueProjections.xml -l HQTarget -p OpsPublisher -i
  "Revenue Projections"
```

The system displays the following:

```
Package "...\\Everserve\\Server\\Packages\\RevenueProjections.xml" with ID
  "Revenue Projections" queued for delivery to "HQTarget" by
  "OpsPublisher".
```

4. Review the status of the delivery using show deliveries and show receipts commands.

```
show deliveries
```

The system displays the following:

Description	Date	Status	Receipts	Targets
deliver -i "Revenue Projections"	07/18/02 17:40:44	Pass	1	1

1 row printed.

```
show receipts -i "Revenue Projections"
```

The system displays the following:

Peer name	Package ID	File name	Status	Received
HQTarget	Revenue Projections	RevenuProjections.xml	Pass	07/18/02 17:40:57

1 receipts printed.

5. View the contents of a return receipt for a target that passed.

```
show receipts -v -i "Revenue Projections" -p HQTarget
```

The system displays the following:

```
Peer name: HQTarget
Package name: Revenue Projections
Specification file: ...\Everserve\Server\Packages\RevenueProjections.xml
Status: Pass
Date received: Thu Jul 18 17:41:01 PDT 2002
Specification:
Status: Pass
Script Return Code: 0
stdout:
stderr:
```

Publishing to a Lists of Targets

Everserve supports package delivery to multiple targets rather than the entire community using the `-l` parameter. The following steps illustrate delivering the package "SecurityPatch.xml" to several targets.

1. Enter the Everserve shell.

```
everserve
```

The system displays the following:

```
Checking Everserve server...
```

```
Everserve 2.0-3.20 Copyright (c) 2001-2002, Synchron Networks, All
Rights Reserved.
```

```
Everserve interactive shell. Type ? or help for a list of commands.
```

2. Deliver the package SecurityPatch.xml to a list of targets from the OpsPublisher system.

```
deliver -f SecurityPatch.xml -l AustinTarget DetroitTarget HQTarget -p
OpsPublisher -i "Security Patch"
```

The system displays the following:

```
Package "...\Everserve\Server\Packages\SecurityPatch.xml" with ID
"Security Patch" queued for delivery to "AustinTarget"
"DetroitTarget" "HQTarget" by "OpsPublisher".
```

3. Review the status of the delivery using show deliveries and show receipts commands.

```
show deliveries
```

The system displays the following:

Description	Date	Status	Receipts	Targets

deliver -i "Security Patch"	07/08/02 17:40:44	Pass	3	3

1 row printed.

```
show receipts -i "Security Patch"
```

The system displays the following:

Peer name	Package ID	File name	Status	Received
DetroitTarget	Security Patch	SecurityPatch.xml	Pass	07/08/02 17:40:59
HQTarget	Security Patch	SecurityPatch.xml	Pass	07/08/02 17:40:57
AustinTarget	Security Patch	SecurityPatch.xml	Pass	07/08/02 17:40:56

3 receipts printed.

4. View the contents of a return receipt for a target that passed.

```
show receipts -v -i "Security Patch" -p AustinTarget
```

The system displays the following:

```
Peer name: AustinTarget
Package name: Security Patch
Specification file: ...\Everserve\Server\Packages\SecurityPatch.xml
Status: Pass
Date received: Mon Jul 08 17:41:01 PDT 2002
Specification:
Status: Pass
Script Return Code: 0
stdout:
stderr:
```

Everweb Operations

The following sections describe the tasks used to manage communities using Everserve's Web-based graphical user interface, Everweb. The information provided describes how to use Everweb to perform specific operational tasks only and does not discuss the rules or policies supporting community and peer definitions. For this information, refer to the section [Community Management](#) in this guide.

The topics in this section include:

- [Connecting to Everweb](#)
- [Everweb Page Overview](#)
- [Managing Communities](#)
- [Managing Peers](#)
- [Viewing System Log Files](#)
- [Server Operations](#)
- [Logging Off Everweb](#)

Connecting to Everweb

Everserve's Web interface (Everweb) requires cookies to be enabled for the session. If a browser is set to refuse cookies, Everweb will let you login only; you will not be allowed to make any selection or perform any task.

If using Internet Explorer, you can enable cookies for "Local Intranet" zone, per session only. Other cookie settings can be disabled if you are administering the local machine only. For Netscape (6.2), cookies can be set to "Enable cookies from originating site only."

Note: If at install time you choose to install Everweb using HTTPS, a Security Alert dialog box will be displayed when invoking Everweb. This alert prompts you to accept the certificate issued by Synchron Networks. To by pass this alert for future sessions, it is recommended that you obtain a valid certificate by an authorized certificate authority.

To Connect to the Everweb User Interface:

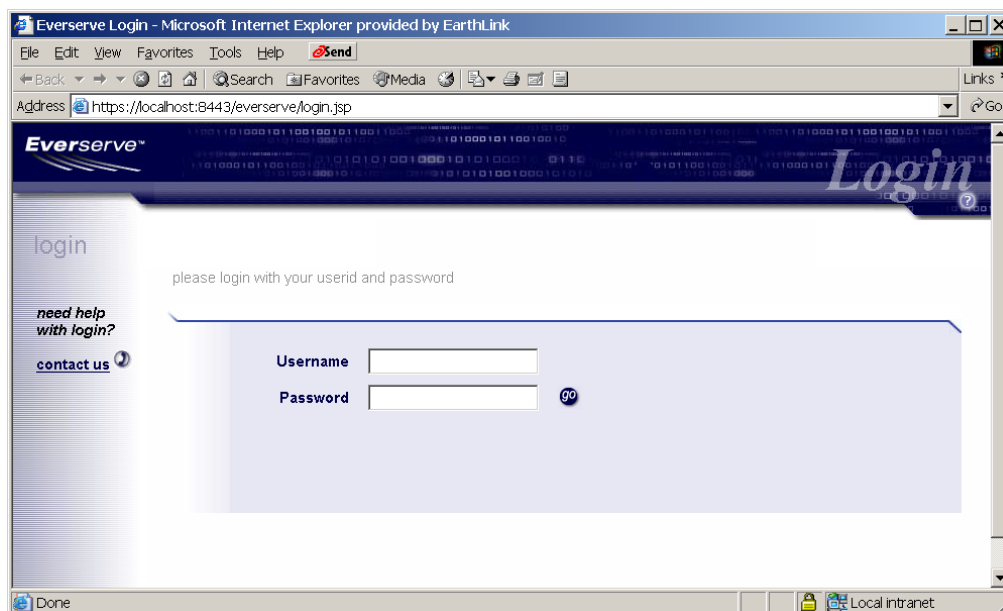
Windows:

From the Windows Start Menu, choose **Programs> Everserve> Everweb Session**.

Solaris:

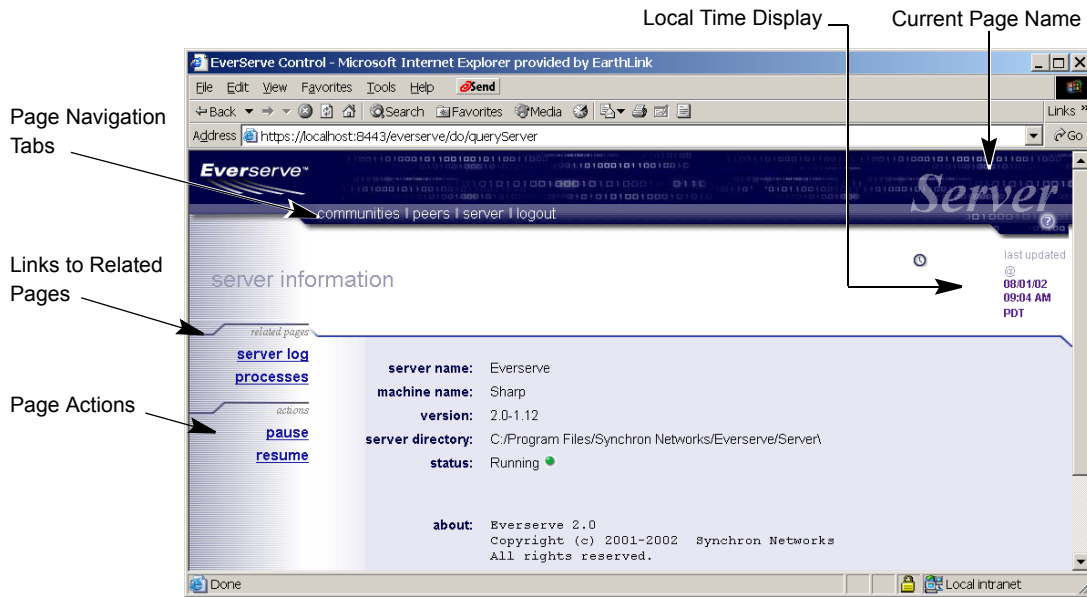
From the command line, type **everweb**.

Once connected to Everserve's graphical user interface, the system displays Everserve's Login Web page:



1. Enter your system login name and password and click **Go**.

The system displays the Server page:



Everweb Page Overview

The top pane of the Web page displays the navigational tabs used to traverse through Everweb, the local time for the device, and the current page name.

Note: The system will display navigational tabs for the role capabilities installed on the device only. For example, if the device was installed with publisher capabilities only, tabs that enable package creation and deployment will be displayed, however community management tasks will not be available.

The left pane contains links to task related pages and a list of actions that can be performed on the current page. The center of the page is used to display or enter information.

Page Navigation

Everweb contains tabs at the top of the page used to navigate between pages. Clicking once on a tab displays the selected Everweb page. Available page navigation tabs are based on role capabilities installed on the device and may include:

Tab	Page Description
Communities	The Communities page is used to view, create, and maintain community definitions.
Peers	The Peers page is used to view, create, and modify peers, and to add peers to a community. The role a peer has in the community is specified while adding the peer to a community.
Packages	The Packages page is used to view, create, and modify package specifications.
Deliveries	The Deliveries page is used to initiate a package delivery, monitor the progress of deliveries, and to view return receipts.
Server	The Server page is used to access system log files, process logs and activity records, and to pause and resume the local device.
Logout	The Logout page is used to log out of the Web interface only - logging out does not disrupt your Everserve service.

Status Indicators

Several Everweb pages contain colored lights to indicate the status of the system or the status of a delivery. The status indicators represent the following conditions respective to the active page:

Indicator Color	Server Page	Deliveries Page
Green	The Everserve system is running.	A green light indicates all expected receipts have been received, and all packages have been successfully delivered and executed without error.
Yellow	The Everserve system is in a paused state.	Indicates that a delivery is still in progress. The total number of return receipts expected for the particular delivery is also displayed.
Red	The Everserve system is not running.	Indicates an error in the delivery for one or more return receipts.

Managing Communities

At install time, the device designated to be a community manager is enabled with capabilities and privileges to create and maintain community and peer definitions. The following sections describe how a System Administrator uses Everweb to define communities, community peers, and package routing information.

All tasks described in this section can be performed from Everserve's command line interface (see the section [Command Line Interface Operations](#) in this user's guide).

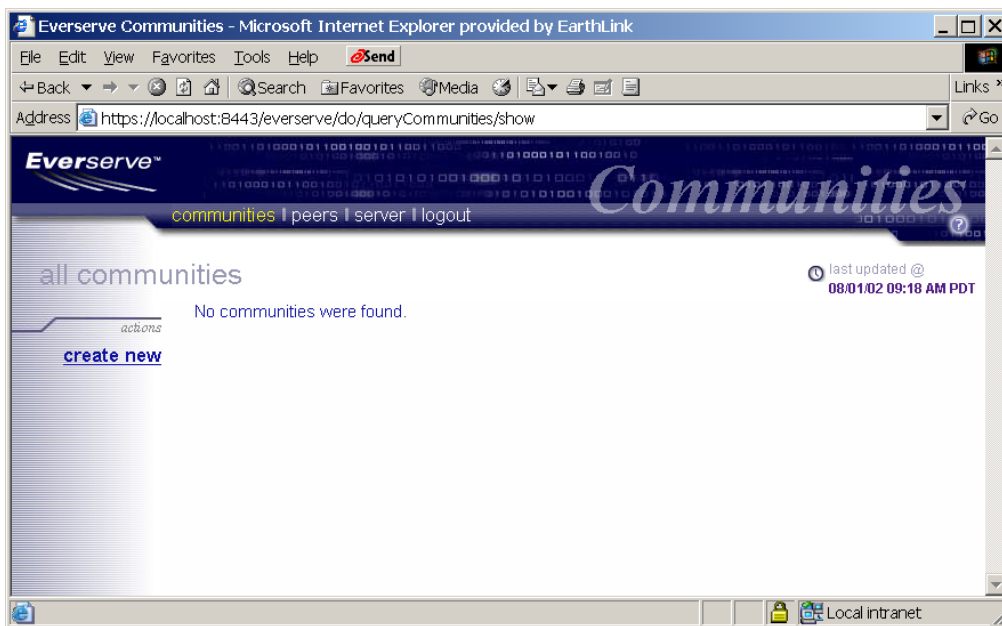
Creating Communities

The community manager peers is automatically created and joined to a community when the community is created.

To Create a New Community:

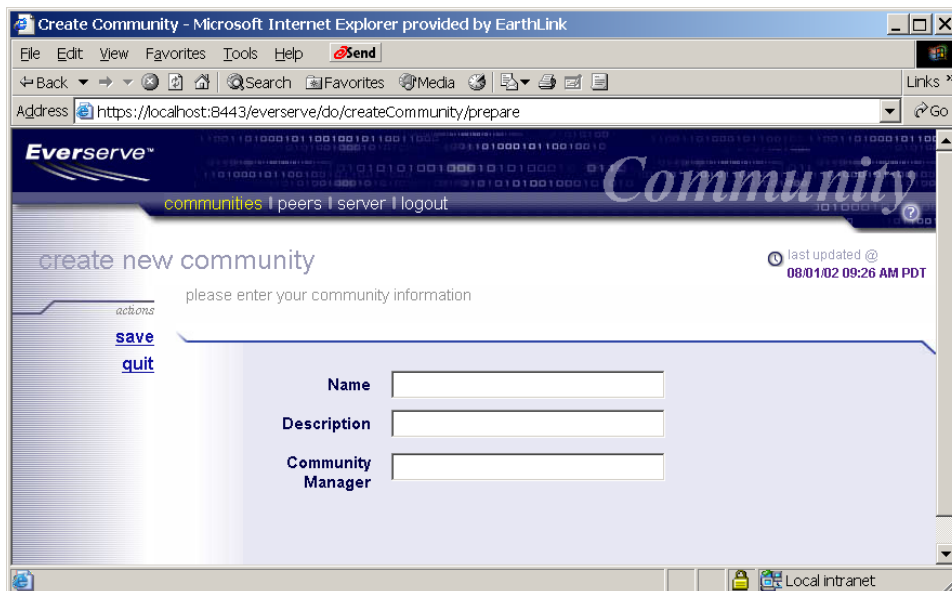
1. Click the **Communities** tab.

The system displays the Communities page. The following example shows no communities exist that this community manager is a member of.



2. Click **Create New**.

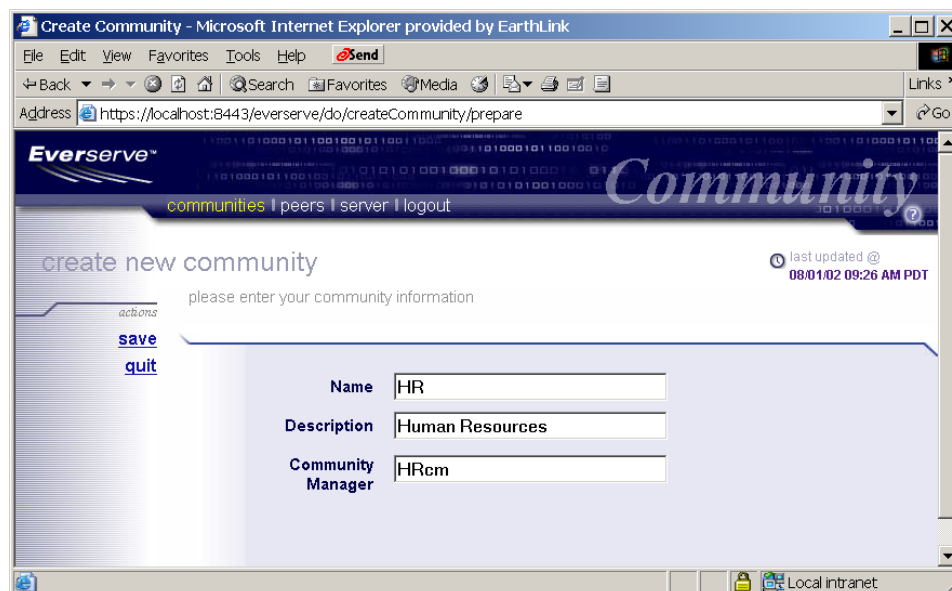
The system displays the following page:.



The screenshot shows a web browser window titled "Create Community - Microsoft Internet Explorer provided by EarthLink". The address bar displays "https://localhost:8443/everserve/do/createCommunity/prepare". The page features the Everserve logo and a navigation bar with links for "communities", "peers", "server", and "logout". The main heading is "create new community", and a subheading says "please enter your community information". On the left, there are links for "actions", "save", and "quit". The form contains three input fields: "Name", "Description", and "Community Manager". A timestamp in the top right corner indicates "last updated @ 08/01/02 09:26 AM PDT".

3. Enter a name and description for the community, and a descriptive name for the community manager.

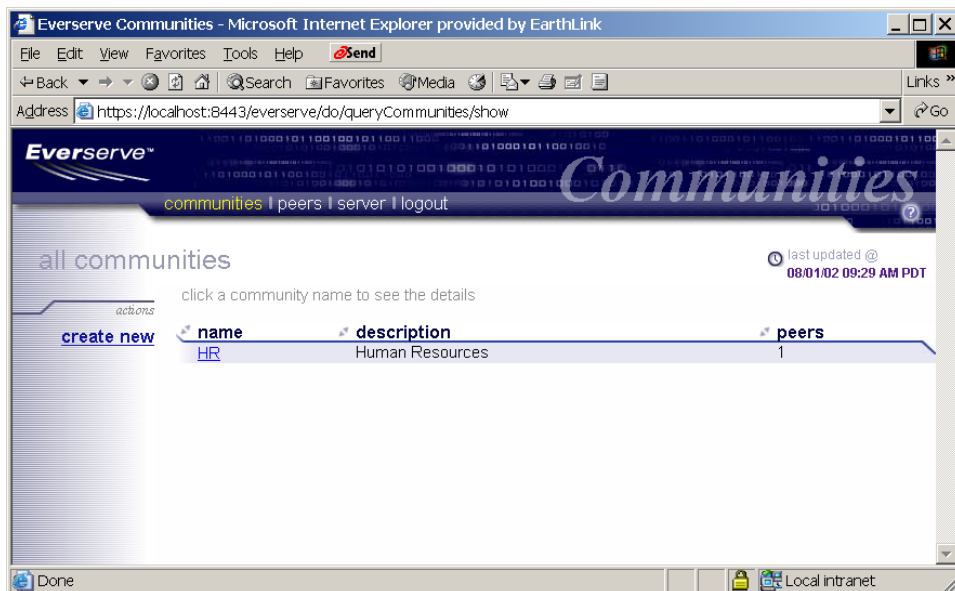
The following screen shows an example of creating a community called "HR" with a description of "Human Resources" and the community manager peer name of "HRcm."



This screenshot shows the same "create new community" page as the previous one, but with example data entered into the form fields. The "Name" field contains "HR", the "Description" field contains "Human Resources", and the "Community Manager" field contains "HRcm". All other elements, including the browser window title, address bar, and navigation links, remain the same.

4. Click **Save**.

The system adds the new community to the list of all other communities this device (community manager) has access to, as shown in the following example:



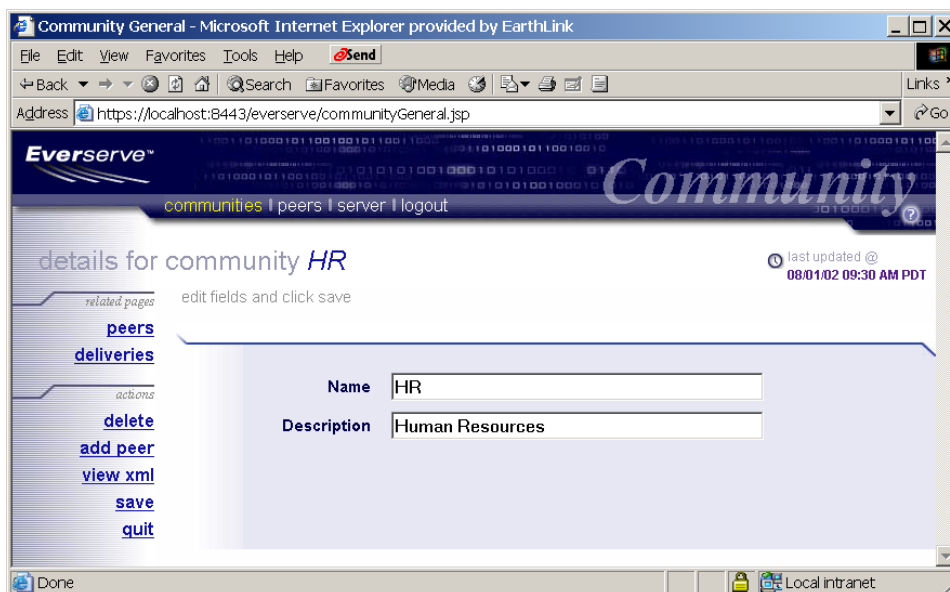
Changing Community Definitions

Using Everserve, you can change the name and description of a community. The following section describes how to change a community's definition.

To Change a Community Definition:

1. From the Communities page, click the **Community Name** you want to change from the list of communities displayed.

The system displays the details page for the community chosen



2. To change the community name, enter a new name for the community in the **Name** field.
3. To change the community description for the community, enter a new description for the community in the **Description** field.
4. Click **Save**.

The system displays the Communities page, listing all known communities.

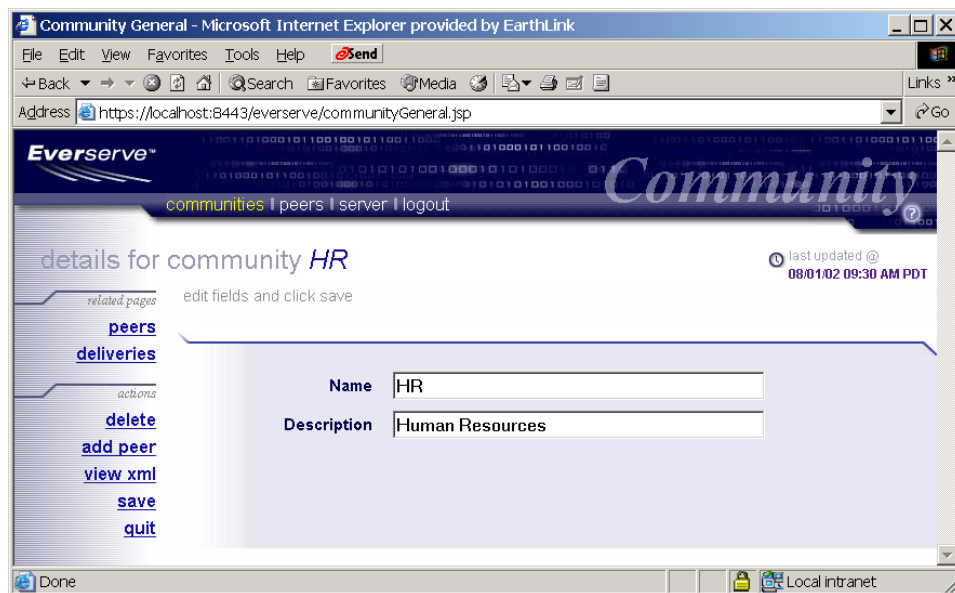
Deleting Community Definitions

As network topologies dynamically change, it may be necessary at time to delete obsolete community definitions from the system. It is important to note that deleting a community definition does not delete any peer definitions from your Everserve system. All peer definitions remain intact and can be included into other communities as needed.

To Delete a Community Definition:

1. From the Communities page, click the **Community Name** you want to delete from the list of communities displayed.

The system displays the details page for the community:



Warning: *Everserve will not prompt you to confirm delete operations. Once you click Delete, the community definition is immediately removed from the database.*

2. Click **Delete**.

After deleting the community definition, the system returns to the Communities page.

Managing Peers

Once a community and community manager have been created, the next step to populate the community with peers and assign each peer a role in the community (such as publisher, relay, or target). The following sections describe how to create peer definitions and add those peer definitions to the community.

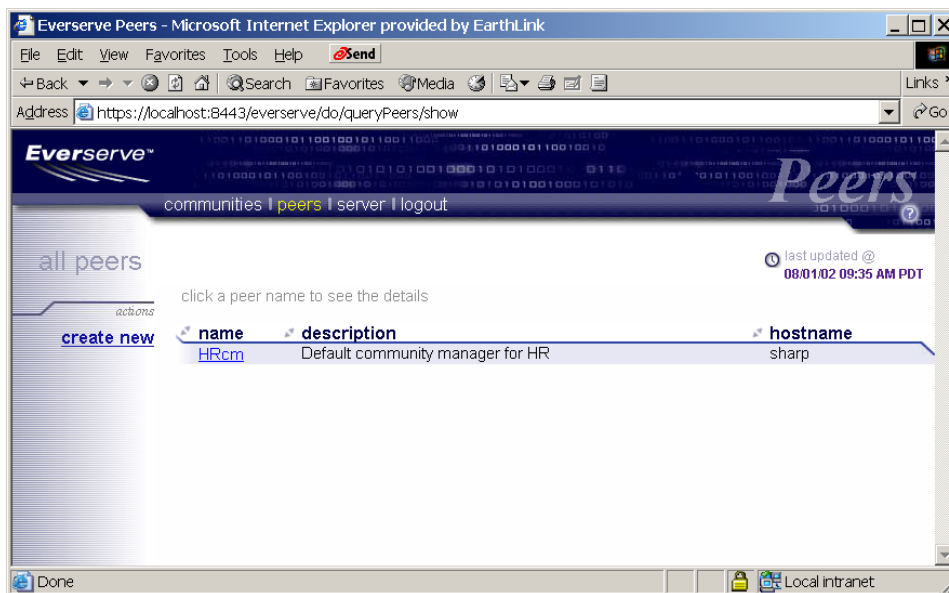
Creating Peers

The community manager can create and modify peers at any time. After creating a new peer and adding the peer to the community, the new peer must obtain the community seed and issue a join command to become an active member and begin receiving messages.

To Create a New Peer:

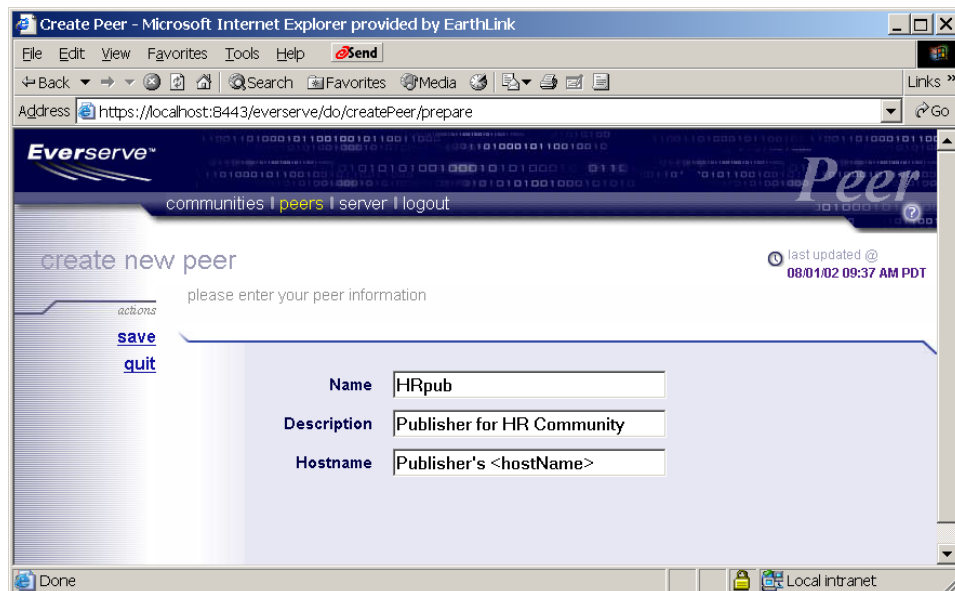
1. Click the **Peers** tab.

The system displays the following screen. Note that the community manager that was automatically created when the community was created.



2. Click **Create New**.

The system displays the following page:

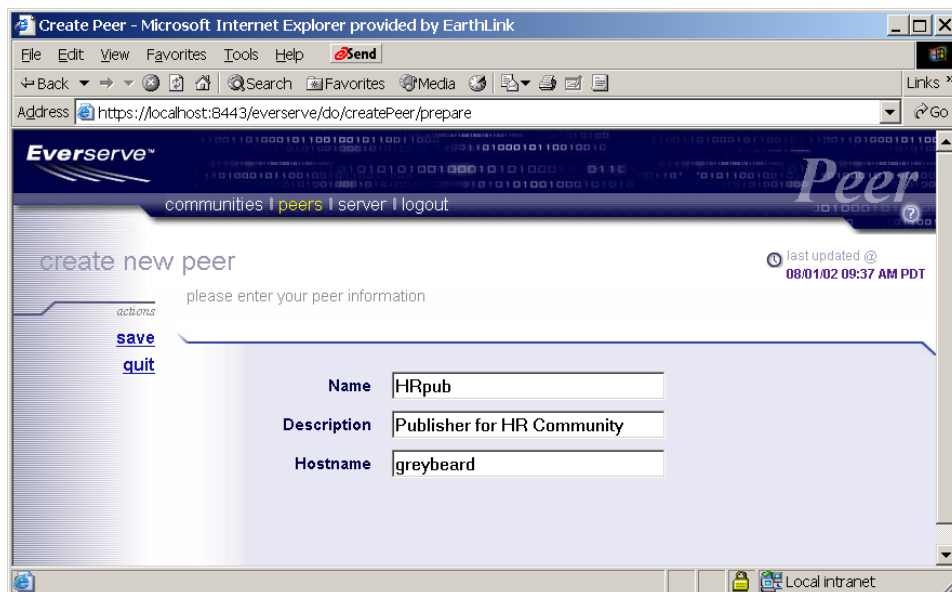


3. Enter the peer name, a description of the peer, and the hostname (physical address) where the peer is located. In the example above, the new peer for the community is given the name "HRpub," a description of "Publisher for HR Community," and the hostname of the publisher device.

Peer hostnames can be a fully qualified hostname or an IP address. The following are examples of valid hostnames:

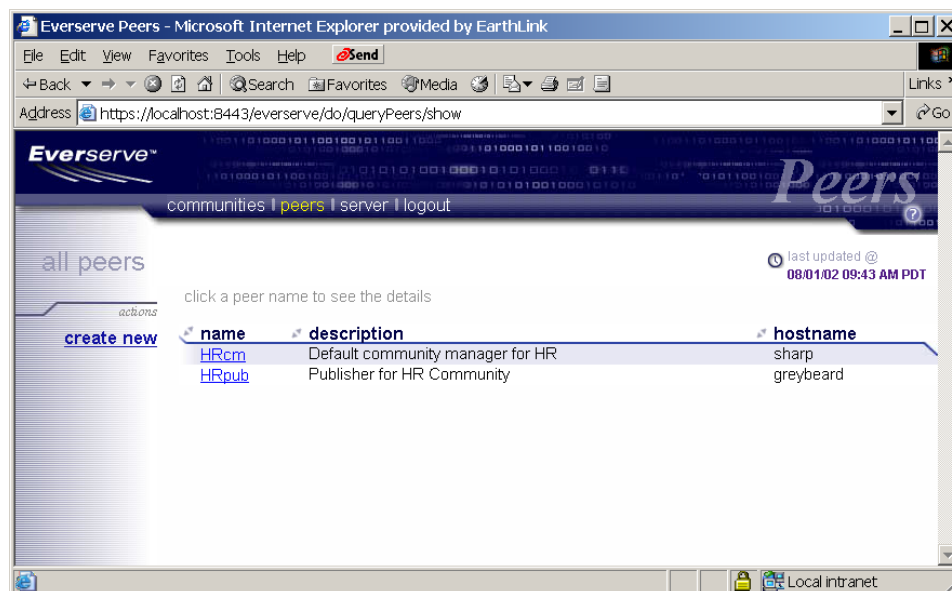
- MyMachine
- MyMachine.example.com
- 012.345.67.89

In this example, the publisher resides on the device (known to the network as) “greybeard.”



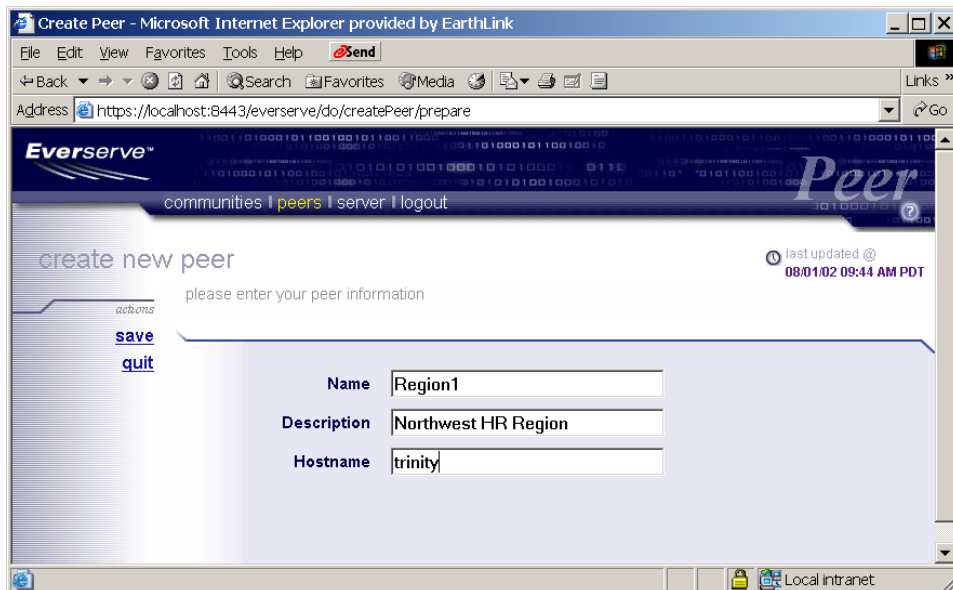
4. Click **Save**.

The system displays the following page, listing the newly created peer:



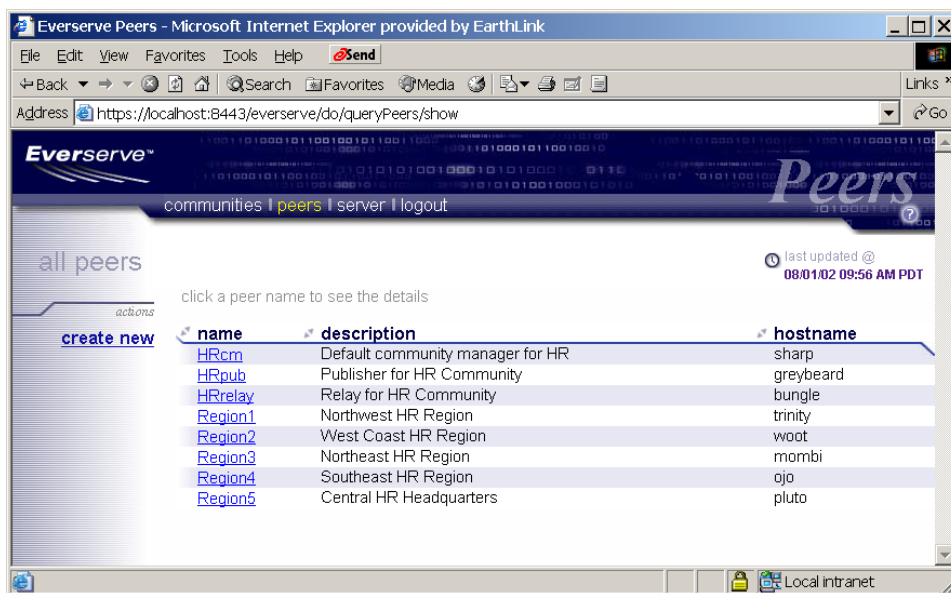
- Repeat steps 2 through 4 to create other peers for the community (such as relays, targets, or additional publishers).

The following example illustrates creating the peer name “Region1,” with a description of “Northwest HR Region.”

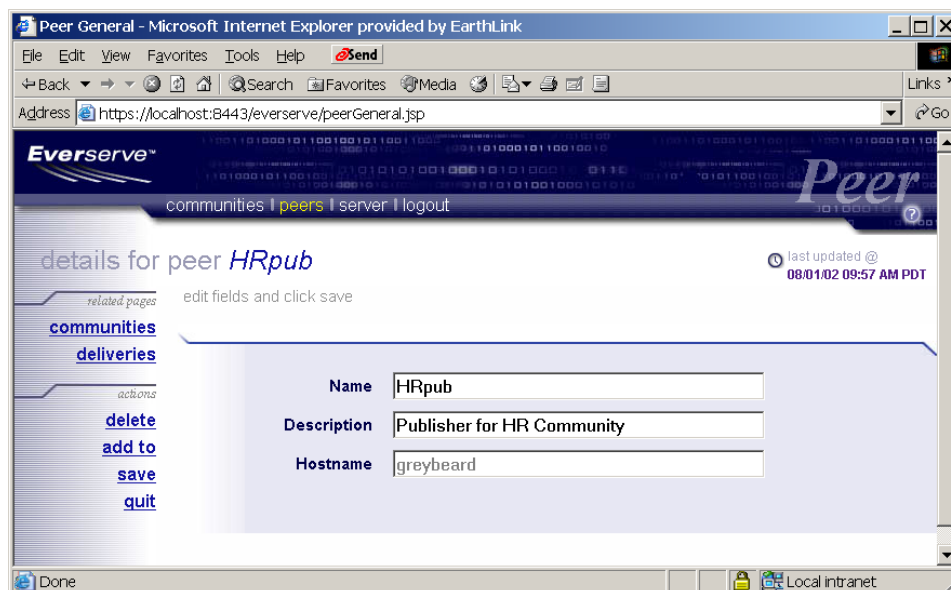


Adding Peers to a Community

1. From the Peers page, click the **Peer Name** you wish to add to a community.

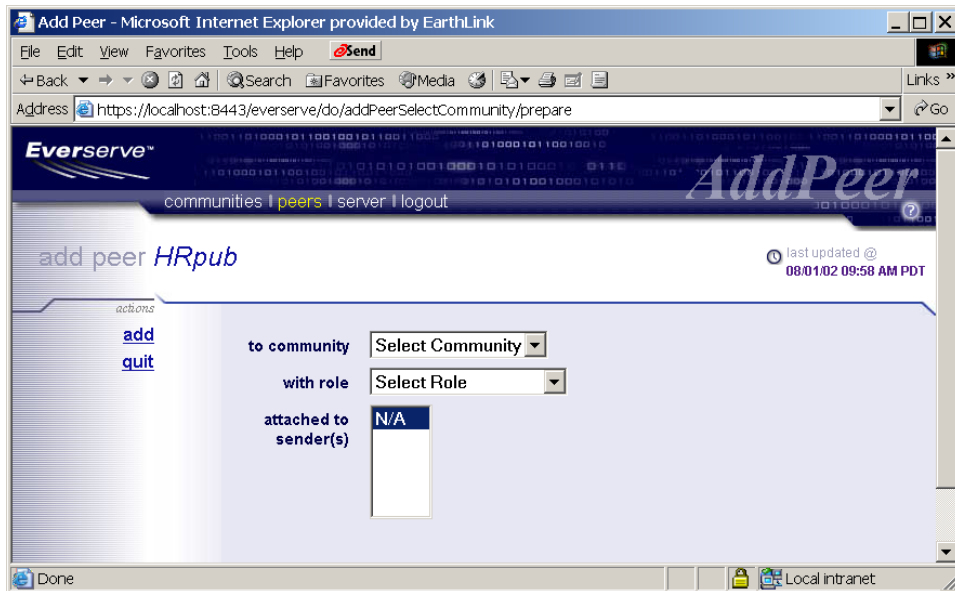


The system displays the following page:



2. Click **Add To**.

The system displays the following page:



3. From the drop down lists, choose the community to which to add the peer, select the role the peer will have in the community, and to which sender the peer is connected.

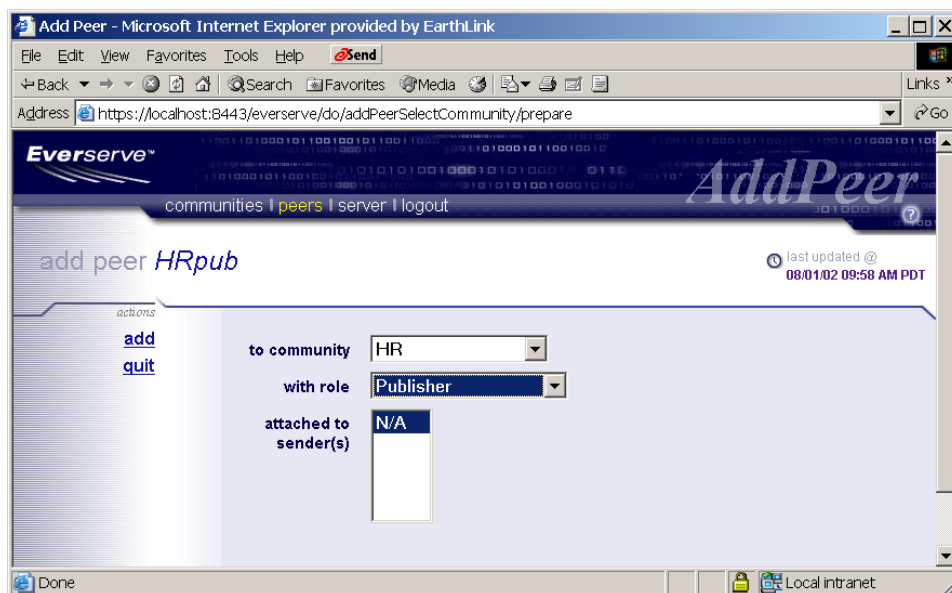
Choosing Peer Roles:

When selecting the role for the peer, select a role that the peer has capabilities of performing. For example, if a peer received a target only installation, assigning a role of relay or publisher will not enable the peer to act as a relay or publisher in the community. To enable additional role capabilities for any peer, you must uninstall Everserve on the peer device, then reinstall Everserve with the desired role capabilities.

Choosing Senders:

Everserve employs specific rules governing peer connections for senders. Generally, all peers must be connected to either *all* publishers in a community, or to one or more relays in a community. For a description of all applicable rules and policies regarding sender/receiver connections, see the section [Specifying Peer Connections in the Community](#).

The following example shows adding the peer “HR pub” to the community “HR” with the role of “publisher.”



4. Click **Add**.

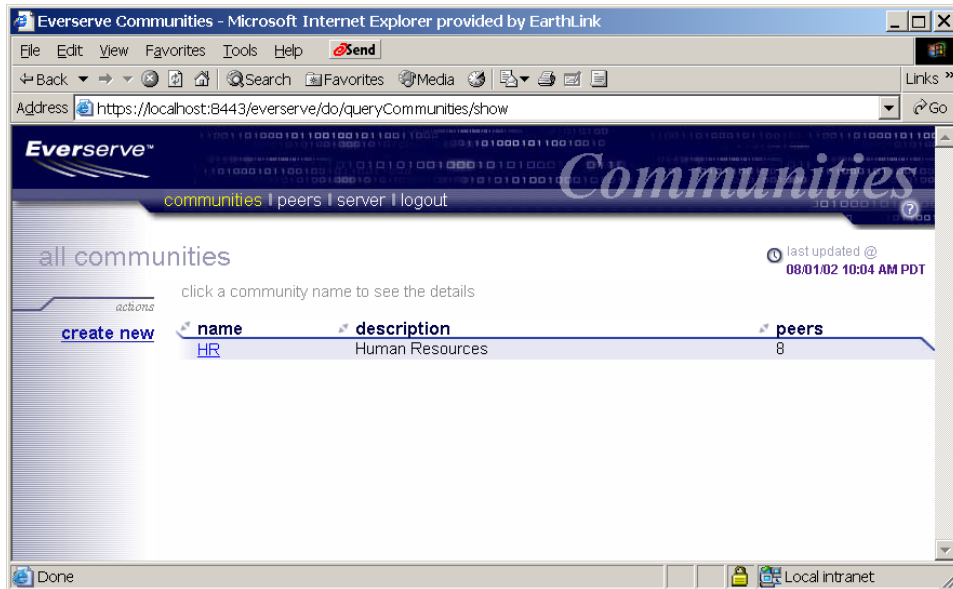
The system adds the peer to the community.

5. Repeat steps 1 through 4 for all other peers you want to add to the community.

Note: All peers added to a community must obtain the community seed file and issue a join command before they can receive packages. Join commands are issued from each peer device in the community and from the command line only. For information on how a peer joins a community, see [Community Topologies](#).

6. Click **Quit** to return to the list of peers for the community.

When you have finished building your community of peers, click the **Communities** tab to view the community information.



This example shows the newly created community "HR," its description, and the number of peers in the community.

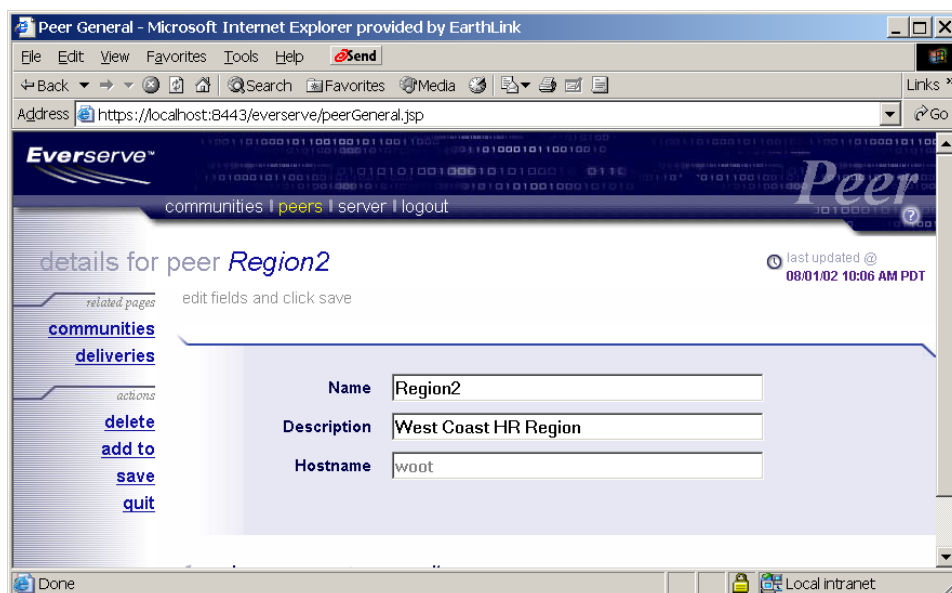
Changing Peer Definitions

Peer definitions can be modified and updated to meet the changing needs of your network topologies. Everserve lets you change the name and description of community peers without interfering with package delivery. However, to change the hostname name of a peer requires that the peer definition be deleted then recreated with the new hostname, then added to the community again. The recreated peer then needs to rejoin the community to become a trusted member and receive packages.

To Change a Peer Definition:

1. From the Peers page, click the name of the peer definition you want to change.

The system displays the following page:



2. Change the peer definition as needed. Enter a new name for the peer or a new description of the peer.
3. Click **Save**.

The system displays the list of peers and their descriptions and hostnames on the Peers page.

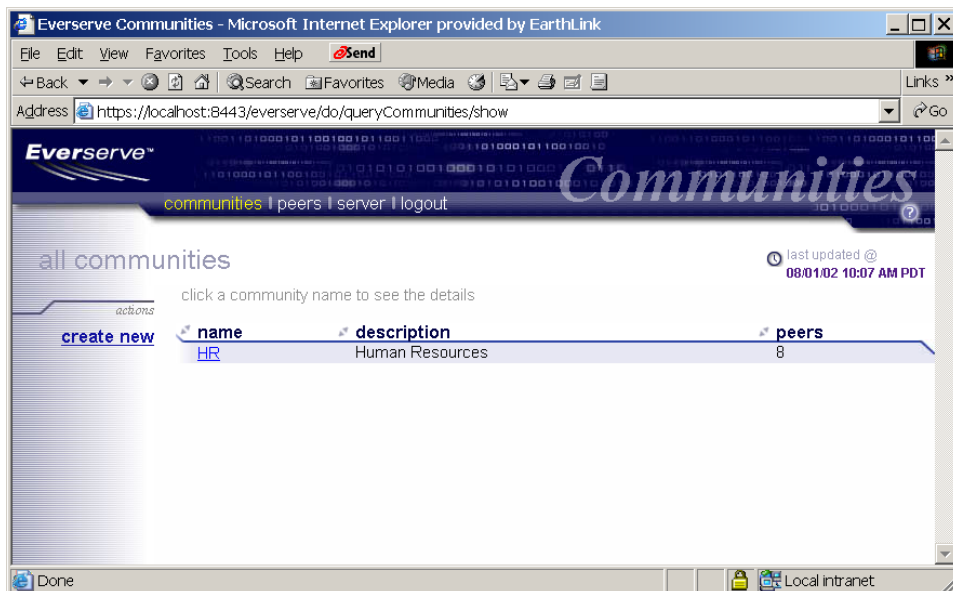
Removing a Peer from a Community

Everserve lets you remove peers from communities without changing or deleting the peer's definition. This is useful if you want to withdraw a peer from a community, disabling the peer from receiving community packages, and to retain the peer's definition for inclusion into an Everserve community at a later time.

To Remove a Peer from a Community:

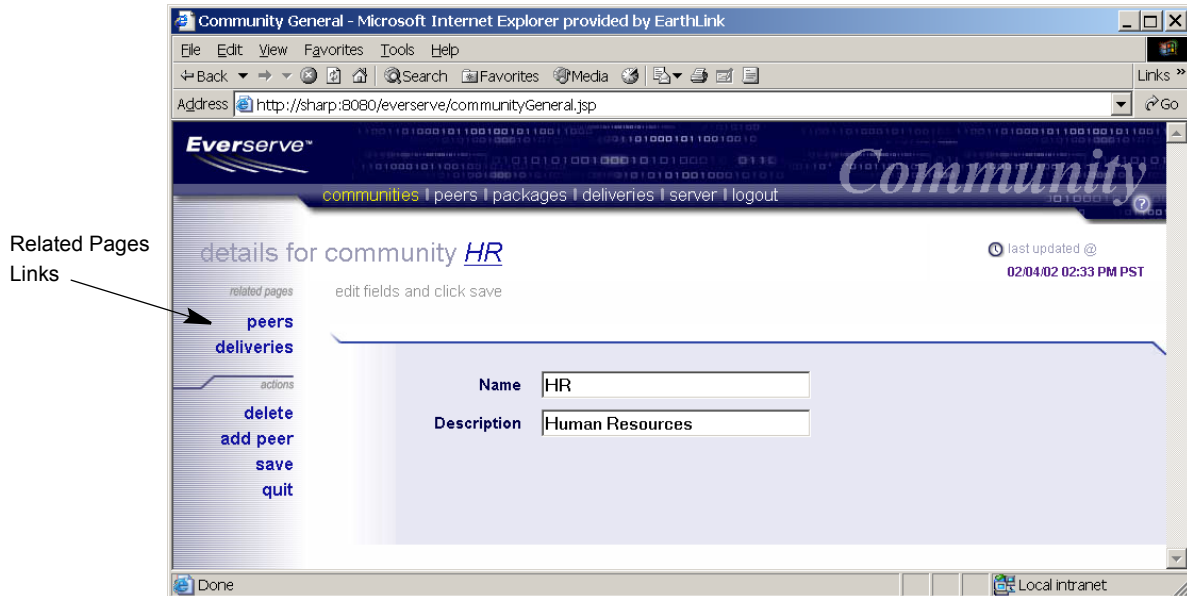
1. Click the **Communities** tab.

The system displays a list of all communities for which the device is either a community manager or member.

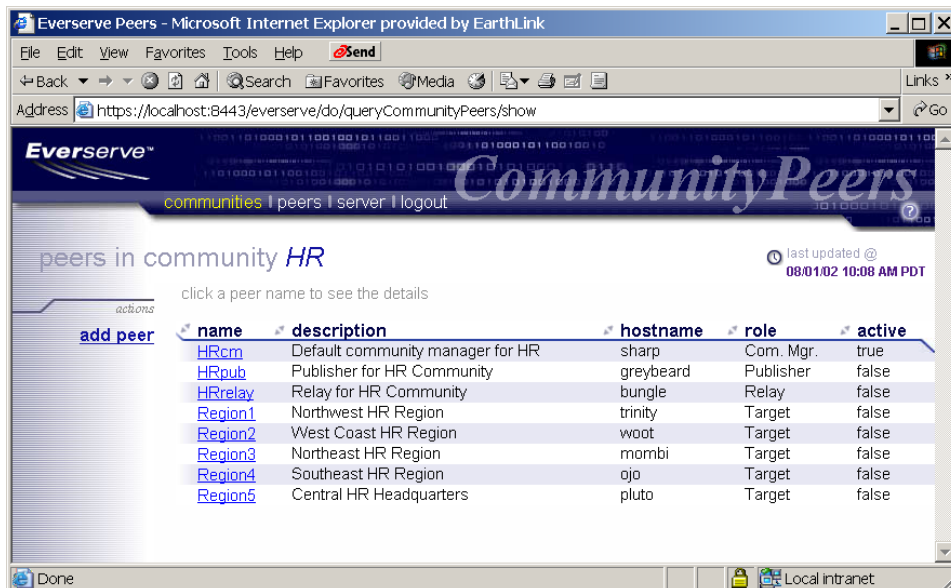


2. Click the **Community Name** that contains the peer you want to remove.

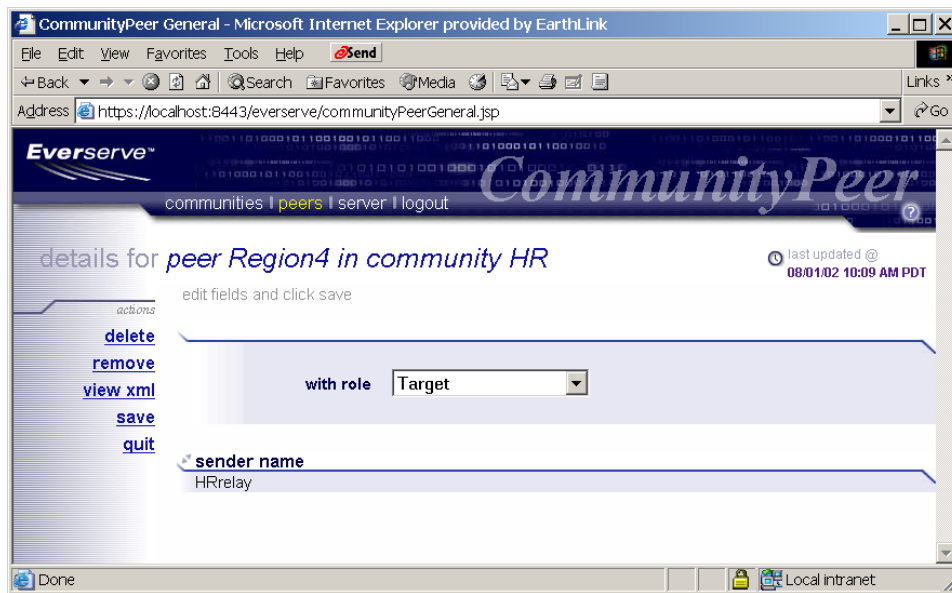
The system displays the details for the community.



- Click **Peers** from the related pages link on the **left side** of the Community page.
The system displays the Community Peers page:

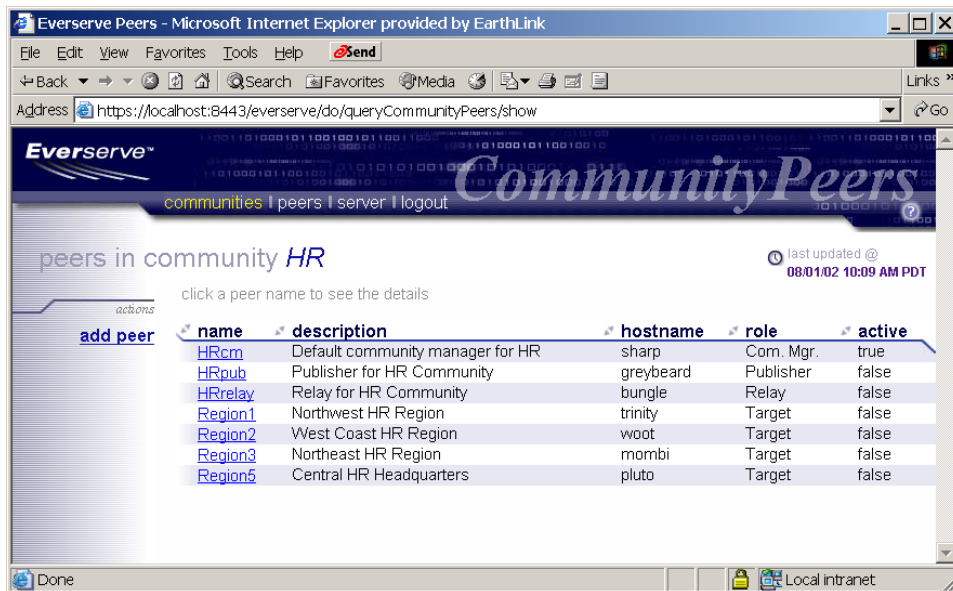


4. Click the **Peer Name** you want to remove from the community.
The system displays the details for the selected peer:



5. Click **Remove**.

The system removes the peer from the community and displays the current list of peers in the community:

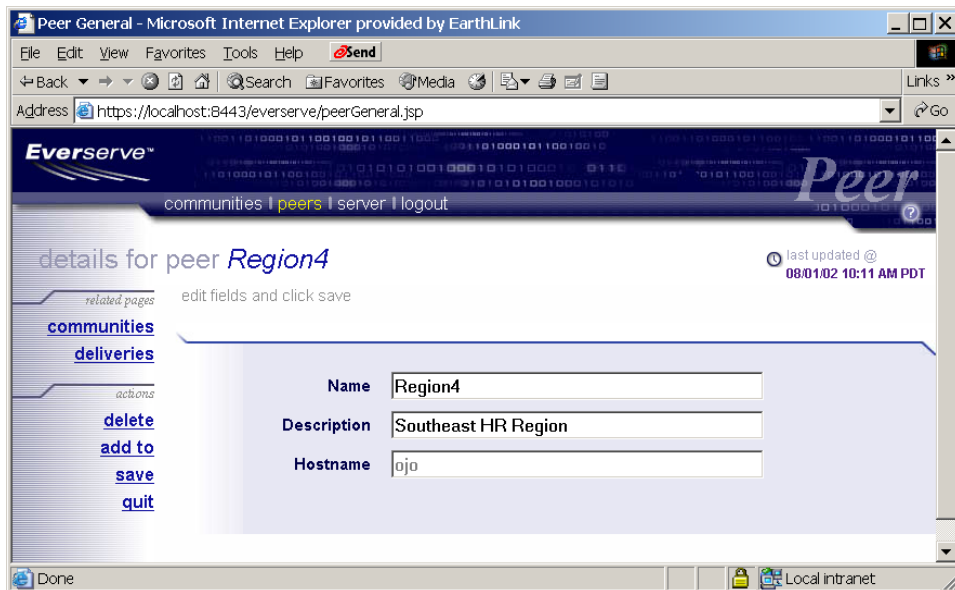


Deleting Peer Definitions

There may be times when it becomes necessary to delete a peer definition from the system. When deleting a peer definition, all information about the peer is removed from the community manager's database.

To Delete a Peer Definition:

1. From the Peers page, click the **Peer Name** you want to delete from the list of peers displayed. The system displays the details page for the peer:



Warning: Everserve will not prompt you to confirm delete operations. Once you click Delete, the peer definition is immediately removed from the database.

2. Click **Delete**.

After deleting the peer definition, the system returns to the Peers page.

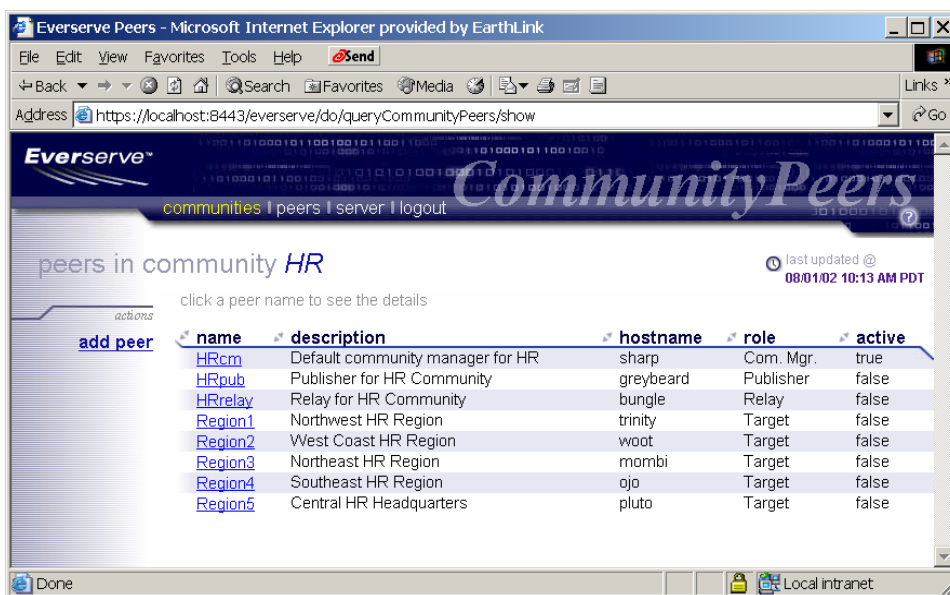
Viewing Community Peers

Once you have created and added peers to a community, you should review the community definition to ensure all peer definitions are correct.

To View all Peer Definitions in a Community:

1. Click the **Community Name** from the list of communities.
2. Click the **Peers** link on the left Actions panel.

The system displays all peers for the selected community, and lists their active status as shown in the following figure:



In this example, the only “active” peers are the ones on the device that created the community - in this example, the community manager. The community manager is automatically activated (joined) because it was created on the Everserve device where the community was created. All other peers will not be active and will not be entitled to receive packages until each peer receives the community seed and joins the community. For information on how to activate community peers, see the section [Community Topologies](#).

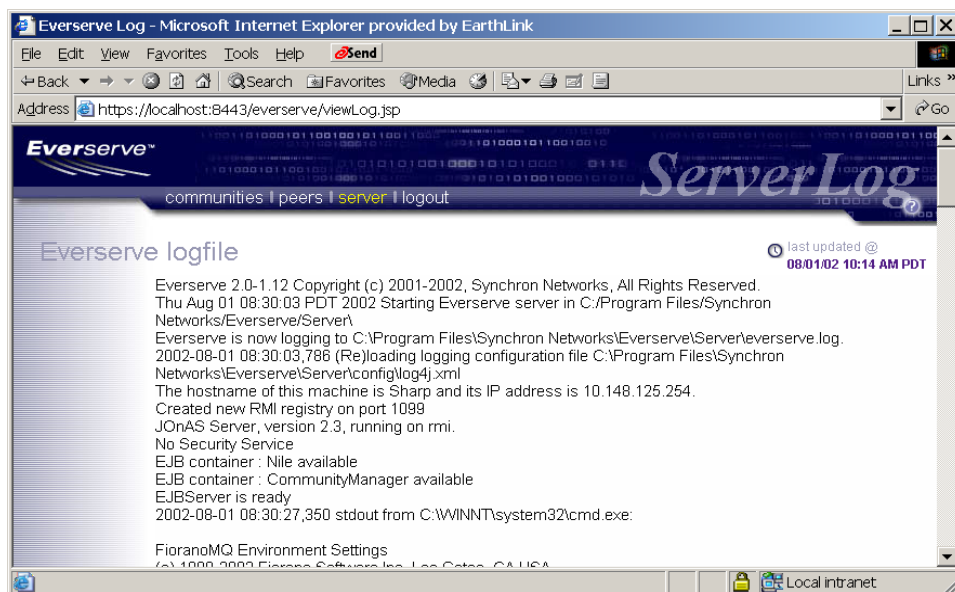
Viewing System Log Files

Everweb provides access to view the system log files (Everserve logs and the process log file) to monitor system activity and events. The following sections provide information on how to access the log files only, for information on the contents and maintenance of Everserve log files, see [System Logs](#) in this guide.

Viewing Server Log File

This file contains information about the server beginning from the last time Everserve was started on the device.

1. Click the **Server** tab.
2. Click the **Server Log** link.
The system displays the Everserve log file.



Viewing the Processes Log File

The process log file can be used to view some or all processes that have been run on the device for a given time period or that match a specified text string. The available viewing filters include:

Filter Type	Description
Predefined Time/Date Range	<p>View the processes log using one of the predefined date/time ranges provided. Select the date range in which to view Everserve processes on the device from the Date pull-down list. Date range options are as follows:</p> <ul style="list-style-type: none">• Past hour• Past 6 hours• Past 12 hours• Past day (24 hours)• Past 3 days• Past week• Past month• Specified (used in conjunction with Start and End date range)

Filter Type	Description
Start and End Date Range	<p>Specify date ranges in which to view processes on the device. The following rules are applicable to date range usage:</p> <ul style="list-style-type: none">• The date parser is locale specific and uses the short date format. In the U.S. locale, the short date format is either mm/dd/yy or mm/dd/yyyy. Leading 0's are permitted, and years may have either 2 or 4 digits.• The display output shows a date and a time, however, the parser only accepts a date and not a time.• If a start date is not specified, the default is the beginning of the current day.• If an end date is not specified, the default is the precise millisecond at which the filter was processed.• You may specify either start and end dates, or you may specify either a start or an end date.• The start date must be less than the end date, regardless of whether the dates are explicit or defaulted.
Text	<p>Descriptive text for an intended process. For example, entering the text string "stop" would result in a display of all Everserve processes that were stopped during the date range specified.</p>

3. After selecting the desired viewing filters, click **Apply Filters**.

The system displays a log file of all Everserve processes and tasks that match the view filters criteria selected. For example:

Everserve Processes

communities | peers | server | logout

all processes last updated @ 08/01/02 10:17 AM PDT

click a process description to see list of activities

actions
[apply filters](#)

filters
date
 specified
 start
 end
text

description	start date	duration	activities
show processes	07/31/02 11:02 AM	0 ms	1
show processes	07/31/02 11:02 AM	0 ms	1
show processes	07/31/02 11:18 AM	0 ms	1
show peers -c "HR" -r Target	07/31/02 01:20 PM	0 ms	1
show peers -c "HR" -r Relay	07/31/02 01:20 PM	0 ms	1
show communities	07/31/02 01:20 PM	0 ms	1
show peers -c "HR" -h "Sharp" -r Publisher	07/31/02 01:20 PM	0 ms	1
show deliveries -s 07/31/02 -e 07/31/02	07/31/02 01:25 PM	0 ms	1
show deliveries -s 07/31/02 -e 07/31/02	07/31/02 01:45 PM	0 ms	1
show deliveries -s 07/28/02 -e 07/31/02	07/31/02 01:45 PM	0 ms	1
show communities	08/01/02 09:17 AM	0 ms	1
show peers -c "HR" -r Target	08/01/02 09:17 AM	0 ms	1
show peers -c "HR" -h "Sharp" -r Publisher	08/01/02 09:17 AM	0 ms	1

Local intranet

Server Operations

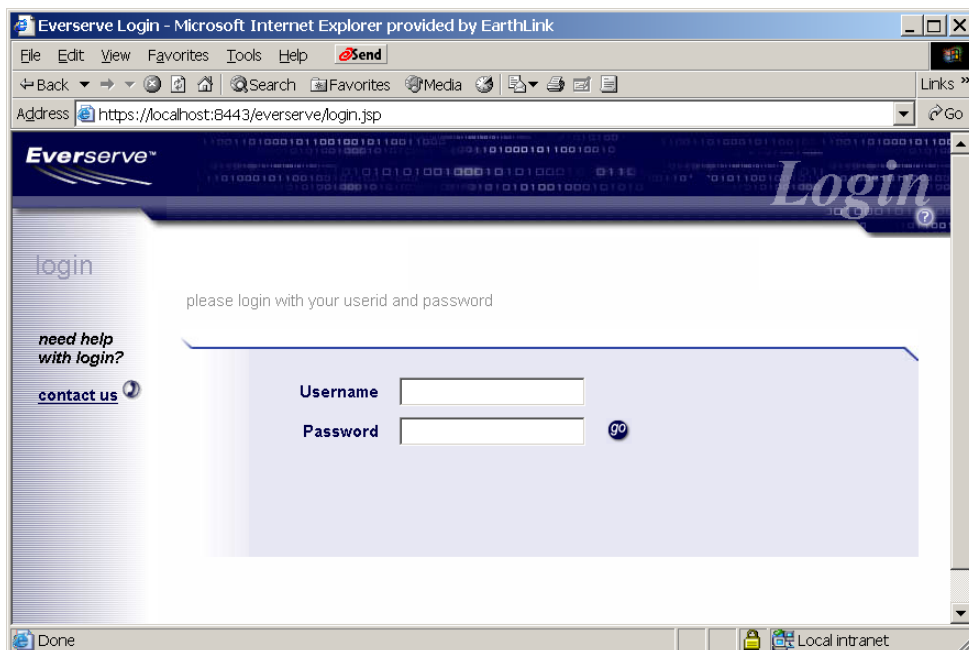
The Server page provides access to Pause and Resume operations. Pausing an Everserve device will disable the device from delivering packages and receiving return receipts. However, any Everserve messages en route will be held in queue until Everserve is resumed on the paused device. To pause or resume Everserve operation, click the desired action listed on the Server page.

When Everserve is paused, the system status light on the Server page turns yellow to indicate a paused state. This light turns green once Everserve has resumed operational status. The following figures show the operational indicators for a paused and running (resumed) operational state. Yellow Status Indicator Showing a Paused State

Logging Off Everweb

To logout of Everweb, click the **Logout** tab.

The system logs the current user out of Everweb and displays the user login screen.



Part 3

Maintenance and Troubleshooting

Database and Filestore Maintenance

Everserve community managers, publishers, and relays maintain their delivery, receipt, process, and activity records in a database, while targets maintain their persistent data on their local file system. In the context of Everserve, persistent data is considered to be any package, community definition, and system audit data.

Since Everserve creates records of all transmissions sent, received, and processed, it is possible to experience performance problems when executing complex queries as databases and filestores reach their maximum record space capacity. Depending on the frequency and amount of data stored by Everserve devices, it will become necessary to perform periodic record maintenance (such as archiving and purging records) for Everserve systems. The frequency in which records are archived and/or purged depends entirely on the amount of disk space used to store Everserve records.

Everserve supports record archiving, purging, and restoration from the command line interface only. The topics in this section include:

- [Database Tables](#)
- [Archiving Records](#)
- [Purging Records](#)
- [Restoring Archived Records](#)
- [Error Recovery](#)
- [Backing-up and Restoring Databases](#)

Database Tables

Everserve logs all process, activity, delivery information for the device into Everserve database tables. These tables may need to be archived periodically to accommodate the continuous information logged to the database tables.

Table Name	Field Name	Description
peer		A Peer is a logical identifier of a device. An Everserve device could have one or more peer identifiers.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	name	The name identifier for the peer. The end user identifies a peer using this name.
	description	An optional description for the peer.
	server_name	An optional Everserve server name that this peer identifies. Especially important if there are more than one everserve server running on the machine.
	host_name	The host name identifier for the peer. Used primarily in the challenge-response protocol of the join operation.
	domain_name	An optional domain name of the peer.
community		A community identifies a grouping of peers. A peer has roles depending on which community it belongs to.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	name	A name identifier for the community. The end user identifies a community using this name.
	description	An optional description for the community.

Table Name	Field Name	Description
community_peer		An associate relationship between community and peers. Used to link peers to a community and vice versa.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_peer	Peer oid.
	oid_community	Community oid.
	active	Whether this peer is active within this community.
key_pair		Stores the public/private key pair for Community peer and Community objects.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_object	Either the community object oid or the community peer oid that the key pair refers to.
	private_key	The private key associated with a community or community peer object. If the key-pair represents a community, then it will be populated only if the community has a local Community manager. If the key-pair represents a community-peer then it will be populated if the peer is local.
	public_certificate	The public signed certificate for the community or community peer.
local_peer		The local peer houses peer identifiers that are local to the device running Everserve.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_peer	The peer identifier.

Table Name	Field Name	Description
transport		The transport entity identifies the underlying transport mechanism used by a peer to send packages to multiple receivers.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_sender	The community peer oid of the package sender.
	roles	The role of the package sender. Possible role values are: 1-Publisher 2-Community Manager 4-Target 8-Relay
	properties_send	A collection of name-value pairs that identify the send properties for the transport. The properties are used to configure the underlying transport (currently JMS).
	properties_receive	A collection of name-value pairs that identify the receive properties for the transport. The properties are used to configure the underlying transport (currently JMS).

Table Name	Field Name	Description
transport_receiver		A transport could have one or more receivers. This table identifies the receivers of a transport.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_transport	The transport that this receiver receives packages from.
	oid_receiver	The community peer identifier of the receiver.
	roles	The roles of the receiver. The role will be moved to the community_peer table in a future release. Possible role values are: 1-Publisher 2-Community Manager 4-Target 8-Relay

Table Name	Field Name	Description
versioned_file		A versioned file houses file attributes of files and directory when it was last scanned for changes. It is used to determine whether a file has to be bundled when version in a package is set to true.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_parent	The oid of the parent
	package_name	The package name that identifies the package specification.
	name	The name of the file or directory.
	size	The size of the file when it was last scanned.
	mode	The permission attributes of the file when it was last scanned.
	atime	The access time of the file when it was last scanned.
	mtime	The modification time of the file when it was last scanned.
	ctime	The change time of the file when it was last scanned.
	shortcut	The path name to which a symbolic link points to. Will be populated only if the file was a symbolic link when it was last scanned.

Table Name	Field Name	Description
package		A package identifies a delivery that was bundled and delivered to targets using the deliver command.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	description	The description of the package. Currently this is filled in with the CLI usage of the deliver command for example: "deliver -i abc -f test.xml -c C1 -p pub".
	oid_process	The oid of the process associated with this delivery.
	specification	The specification xml that was used to bundle this package.
	id	An identifier for the delivery. It is the value of the -l option in the deliver command.
	number_targets	The number of targets that will receive this delivery.
	ts	The time when the package was delivered.
	ts_msec	Similar to ts, but with millisecond accuracy. Time fields in some database like mysql have accuracy only up to a second.
	number_receipts	The number of targets that have acknowledged receiving this delivery.
	status	Aggregated status of the delivery. 0 means success while 1 means failure. It will be set to failure if any of the targets sends a failure receipt.
	oid_community	The community that received this delivery.

Table Name	Field Name	Description
package_transport		The transport used to send a package. It is possible that a package is delivered to targets using multiple transport protocols. Though it is populated currently, it will be deprecated in a future release as its use is limited and the information is redundant.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_package	The oid of the package.
	oid_transport	The oid of the transport.

Table Name	Field Name	Description
package_element		This table contains individual file, directory and command elements that make up a package.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_package	The oid of the package that this element is a part of.
	oid_parent	Identifies the parent package element. All package elements barring the top level package container has a parent package element. This field will be null for the top level package container.
	name	The name of the element.
	type	The package element type. Possible values are: 1 - Package container 2 - Directory 3 - File 4 - Command
	sequence	The In-order traversal sequence number of this element in the package tree.
	data	The data of this package element. Currently the option to populate this field has been turned off as bundling and sending of a package is done together in one command.
	has_changed	Internal to Everserve processing.

Table Name	Field Name	Description
file_element_attributes		This table contains the file and directory attributes of the files that are bundled. Currently the option to populate this field has been turned off as bundling and sending of a package is done together in one command.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_package_element	File or directory package element oid that this record ties to.
	operation	Operation that will be executed on the target. Possible values are: 1- Create 2- Meta 3- Remove 4- No Change 5- Replace
	size	The size of the file.
	mode	The permissions for the file or directory.
	atime	The last access time for the file or directory.
	mtime	The last modification time for the file or directory.
	ctime	The last change time for the file or directory.
	shortcut	The symbolic link that the file points to if the file is a symbolic link.

Table Name	Field Name	Description
receipt		This table houses the acknowledgment sent by a target after unbundling a delivery. There will be one receipt for each target that the package was sent to.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_package	The oid of the package for which this receipt was generated.
	description	An optional description for the receipt. Currently it is automatically generated by Everserve and contains the name of the target, the package id and the specification file name for example: Receipt for ""target"", package id = ""abc"", package name = ""test.xml""
	oid_community_peer	The oid of the community peer that sent this receipt.
	ts	The time this receipt was received by the publisher. If a receipt has not yet been received this field will be null.
	ts_msec	Similar to ts, but with millisecond accuracy. Time fields in some database like mysql have accuracy only up to a second. Will be null if a receipt has not yet been received.
	status	Status of the delivery on the target. Possible values are: 1 - Success 2 - Failure

Table Name	Field Name	Description
receipt_element		Receipt elements contain information about the status of unbundling a package element on a given target. Note that not all receipt elements are saved. If the parent housing directory was unbundled successfully then the receipt elements housed within that directory are not saved. Similarly if unbundling a package container (top level of the tree) was successful, and the package did not contain any command package elements then only the package container success status is saved. Command receipt elements are always saved irrespective of success or failure as it could contain standard output and standard error data.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_receipt	The receipt oid that this element ties to.
	oid_package_element	The package element oid associated with this receipt element.
	status	Whether unbundling this receipt element was successful. Possible values are: 1 - Success 0 - Failure
	rc	Return code returned by the command execution.

Table Name	Field Name	Description
receipt_element_blobs		Receipt element blobs contains the output of command-spec elements. Every receipt_element associated with a command, will have a corresponding element in this table.
	oid	Primary key. It is identical to the command's receipt element oid.
	stdout	The standard output of the command execution on the target.
	stderr	The standard error of the command execution on the target.
process		This table groups activities that occur at any given time using Everserve. It tracks both synchronous and asynchronous events that occur on everserve. All CLI and Web commands are tracked as individual processes as well as deliveries received by the target.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	description	A brief description of the activity. For CLI commands, the paraphrased version of the actual command issued by the user is saved.
	ts_min	Time when this process began.
	ts_max	Time when the last activity for this process occurred.
	nbr_activities	Number of activities that are currently logged for this process.
	object_type	Currently not used.

Table Name	Field Name	Description
activity		This table houses every notable event that occurred in Everserve. Each activity has an associated process record that groups it.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	oid_process	Oid of process that this activity is tied to.
	description	Description of this activity.
	status	Status of the activity. It is a integer code, whose description is stored in the descriptor table under type =1.
	ts	The time when the activity took place.
	ts_msec	Similar to ts, but with millisecond accuracy. Time fields in some database like mysql have accuracy only up to a second.
	username	The user who initiated this event.
	level	Currently always 2.
	object_type	
	oid_object	

Table Name	Field Name	Description
object_type		This is a static table mapping a type to an Everserve class name. One can construct a live Everserve object, by instantiating the class associated with a type and passing the appropriate oid in the constructor. Everserve follows a simple object relational model and there is a one to one mapping between an everserve class and the database row. It is for this reason that all database tables is a single field (UUID) of the same type.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	type	The type of object. There is a uniqueness constraint on this field.
	class_name	The class name to be instantiated for the given type.
	attributes	Any other attributes that needs to be passed.
	comment	A description text for this object type.
descriptor		This is a static table that maps type and id combination to a description. The type is the broad category while the id are the various identifiers within that category.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	type	Currently there are three categories of descriptors: 1 - status field in the activity table 2 - level field in the activity table 3 - type field in the delivery_log table
	id	Identifier within the type field. There is a uniqueness constraint on the type, ID combination.
	description	Description for the identifier of a given type.

Table Name	Field Name	Description
delivery_received		This object contains deliveries that have been received by a target or a relay. It is used to prevent duplicate processing of the same package through multiple routes.
	oid	A composite UUID that guarantees uniqueness of a package for a given target. We need two UUIDs here as just the delivery oid will not suffice as the package can be received twice if there are more than one target for the same delivery. To guarantee uniqueness for a target, we need to append the listener oid as well to enable us to realize that the packet indeed has been received twice.
	ts_msec	When the package was received as a long since epoch. We do not store it as a time field as databases like MySQL do not provide millisecond accuracy. Besides most targets have file store where this information is cached in memory. In a future release, we will be using this field to clear the cache (for file stores) after a specified interval.

Table Name	Field Name	Description
delivery_log		A log of deliveries that were sent, relayed or received by the Everserve server. This information can also be retrieved from activity records, but would have to be filtered from other types of events that took place. Besides the delivery log houses additional information that are not captured in activity records.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	type	The type field that specifies the action taken. The exact description corresponding to a value is stored in the descriptor table.
	id	The id of the package that was received or delivered.
	ts	When the package was received or delivered.
	ts_msec	When the package was received or delivered measured as a elapsed time in milliseconds.
	process_time	How much time was taken to process the delivery (in milliseconds).
	originator	Originator peer for this package.
	sender	The sender of this package. Could be different from originator if the sender is a relay.
	community	The community to which this package was delivered
	peer	The local peer within the community that was a recipient for this package.

Table Name	Field Name	Description
db_version		Stores version information for this everserve schema. There is only one record in this table. Its use is to provide version compatibility between the everserve schema and everserve server.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space.
	version	The current version of the database schema.
	earliest	The earliest version that this schema is compatible with.
jms_ssl_server_properties		This table houses the Key Pair oids if we are using SSL to communicate with the JMS server. This table is only applicable for peers that are senders (community manager, publisher, and relay). This however poses a limitation that we can only have one Transport Server (JMS) per Everserve server. In a future release we will begin having named transport to get around this.
	oid	Primary key. It is a UUID guaranteed to be unique across time and space
	oid_server_key_pair	JMS uses the key pair to validate client access to queues and topics. An Everserve sender needs the key to sign certificates while creating communities.
	oid_client_key_pair	Everserve sends the Client Certificate and seed over to receivers in a seed file.
	oid_admin_key_pair	Everserve uses this key pair to add or remove JMS queues and topics.

Archiving Records

The **archive** command moves the records associated with a specified set of deliveries, processes or delivery logs from the main database table to a set of secondary database archive tables. This action is similar to moving deliveries, processes, and delivery logs to a separate folder.

The schema for each secondary table is identical to its counterpart Everserve table. The archive tables are housed within the same database for database servers that do not allow joins across databases (for example, MySQL) or in a separate database (everserve_archive) for databases such as MS SQLServer that allows joins across databases. For MySQL, these tables are MyIsam tables; for MS SQLServer, the archive tables are located in a separate database and have the same name as their counterpart in the Everserve database.

Note: Once an archive action is performed, then subsequent activities related to an archived set are lost. For example, the receipt will be lost if a delivery was archived, and a pending receipt is received after the archival has taken place.

Best Practices

Archiving records should be done periodically so that access to information in the database or file store does not deteriorate in speed. The following are recommended practices to archiving records:

- When a delivery or process is no longer pertinent and no longer needs to be observed.
- When show commands become too slow. This may be especially true for the show receipts command.
- Archiving records should be done in manageable chunks. For example, if a years worth of data is being archived, on some databases, it may take a very long time to complete the archival process. If, however, this task is divided into monthly chunks, then the 12 [monthly] operations may take less time to complete than one operation for an entire year.

Specifying Date Ranges

Processes and delivery logs can be archived based on the date range specified using the `-s` (start date) and `-e` (end date) tags as described in the following sections. The following rules are applicable when using start and end dates:

- The date parser is locale specific and uses the short date format. In the U.S. locale, the short date format is either mm/dd/yy or mm/dd/yyyy. Leading 0's are permitted, and years may have either 2 or 4 digits.
- If the `-s` tag is omitted, all records up to the end date will be archived.
- If the `-e` is omitted, all records later than the start date will be archived.
- If both `-s` and `-e` are omitted, all records will be archived.
- If both `-s` and `-e` are specified, all records that fall in the date range will be archived.

Archiving Delivery Records

The **archive deliveries** command is used to store all delivery related tables that match the specified criteria into archive tables.

Command Syntax:

```
everserve archive deliveries [-s <startDate>] [-e <endDate>] [-x]  
[-i <ID>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage
[-x]	N	Excludes all pending deliveries from being archived even if the pending deliveries falls within the specified date range.
[-i <ID>]	N	The ID of the package in which to archive, such as "update_2002." Multiple IDs are separated with a space, for example: -i update_2002 update_2002_HR

Example:

```
$ everserve archive deliveries -s 08/09/2002 -e 09/09/2002 -x  
-i update_2002
```

Archiving Process Records

The ***archives processes*** command is used to store processes and activity tables that match the specified criteria into archive tables.

Command Syntax:

```
everserve archive processes [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage

Example:

```
$ everserve archive processes -s 09/09/2002 -e 09/17/2002
```

Archiving Deliverylog Records

The **archive deliverylog** command is used to store deliverylog tables that match the specified criteria into archive tables.

Command Syntax:

```
everserve archive deliverylog [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage

Example:

```
$ everserve archive deliverylog -s 09/09/2002 -e 09/17/2002
```

Archiving All Records

The **archive all** command is used to archive all records into a separate database or store. The archive all command moves the data from the deliveries, processes, and activity tables into corresponding backup archive tables.

Command Syntax:

```
everserve archive all [-s <startDate>] [-e <endDate>] [-x]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be archived. See Specifying Date Ranges for information of acceptable usage
[-x]	N	Excludes all pending deliveries from being archived even if the pending deliveries falls within the specified date range.

Example:

```
$ everserve archive all -s 09/09/2002 -e 09/17/2002 -x
```

Purging Records

The **purge** command deletes archived records for a given category. This command will only delete records from the archive tables and not the Everserve database. When using the **purge** command, an optional directory output argument may be used that will purge records then move the purged records to the specified secondary storage directory. These records can later be restored using the **restore purged** command. For information on how to restore purged records, see the section [Restoring Archived Records](#).

Best Practices

Purging records should be done periodically to effectively manage archived records. The following are recommended practices to purging records:

- Purge when the archive tables become unmanageable or the performance of the show -a (archive) commands deteriorate.
- For MySQL, there may be an OS limit to the size of the archive database table file. Archive tables for MySQL are MyIsam, as opposed to InnoDB for the main tables.
- If the purged records are stored in a secondary directory, then it is important that some directory-naming standard is employed. One example for naming the directory would be Everserve_All_010102_to_013102. This suggests that the directory houses all deliveries, processes and delivery logs for the month of January 2002.

Note: Each purge operation requires a new directory to house purged records. The purge command will throw an error if attempting to execute a purge command that purges twice to the same directory location.

The following commands are used to purge archived records.

Note: The start and end date range parameters for purging archived records are used in the same manner as described in the section [Specifying Date Ranges](#). However, when using start and end date ranges in purge commands, the criteria specified for purging will match records in the archive table and not the main Everserve table.

Purging Delivery Records

The ***purge deliveries*** command deletes delivery and receipt data permanently from the archive tables. Purged data may be saved to a file, enabling record restore operations at a later time.

Command Syntax:

```
everserve purge deliveries [-s <startDate>] [-e <endDate>]  
[-d <directory>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage
[-d <directory>]	N	The full path and directory where the purged record will reside. Purged files will be created with a.dat extension.

Example:

```
$ everserve purge deliveries -s 09/09/2002 -e 09/17/2002  
-d c:\myDeliveries
```

Purging Deliverylog Records

The ***purge deliverylog*** command deletes delivery log data permanently from the archive tables. Purged data may be saved to a file, enabling record restore operations at a later time.

Command Syntax:

```
everserve purge deliverylog [-s <startDate>] [-e <endDate>]  
                             [-d <directory>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage
[-d <directory>]	N	The full path and directory where the purged record will reside. Purged files will be created with a .dat extension.

Example:

```
$ everserve purge deliverylog -s 09/09/2002 -e 09/17/2002  
  -d c:\myDeliverylogs
```


Purging Process Records

The ***purge processes*** command deletes process data permanently from the archive tables. Purged data may be saved to a file, enabling record restore operations at a later time.

Command Syntax:

```
everserve purge processes [-s <startDate>] [-e <endDate>]  
                        [-d <directory>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage
[-d <directory>]	N	The full path and directory where the purged record will reside. Purged files will be created with a .dat extension.

Example:

```
everserve purge processes -s 09/09/2002 -e 09/17/2002 -d c:\myProcesses
```

Purging All Records

The ***purge all*** command removes delivery, receipt, delivery log, and process data permanently from the archive tables. Purged data may be saved to a file.

Command Syntax:

```
everserve purge all [-s <startDate>] [-e <endDate>] [-d <directory>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the data to be purged. See Specifying Date Ranges for information of acceptable usage
[-d <directory>]	N	The full path and directory where the purged record will reside. Purged directories will be created with a .dat extension.

Example:

```
$ everserve purge all -s 09/09/2002 -e 09/17/2002 -d C:\myDeliveries
```

Restoring Archived Records

The **restore** command is the undo operation to the archive command. It moves archived records back into the main Everserve tables. This is true for all the restore commands except the **restore purged** command. The restore purged command moves the records from the secondary storage back into the archived tables.

Restoring Delivery Records

The **restore deliveries** command is used to retrieve package and receipt data from backup archive tables to the original tables.

Command Syntax:

```
everserve restore all [-s <startDate>] [-e <endDate>] [-i <ID>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage
[-i <ID>]	N	The ID of the package that was archived, such as "update_2002."

Example:

```
$ everserve restore deliveries -s 09/09/2002 -e 09/17/2002 -i  
  update_2002
```

Restoring Deliverylog Records

The ***restore deliverylog*** command is used to retrieve delivery log data from the archive tables to the original tables.

Command Syntax:

```
everserve restore deliverylog [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage

Example:

```
$ everserve restore deliverylog -s 09/09/2002 -e 09/17/2002
```

Restoring Process Records

The ***restore processes*** command is used to retrieve process data from backup archive tables to the original tables.

Command Syntax:

```
everserve restore processes [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage

Example:

```
$ everserve restore processes -s 09/09/2002 -e 09/17/2002
```

Restoring All Records

The ***restore all*** command is used to retrieve data from backup archive tables and restore this data to the original tables.

Command Syntax:

```
everserve restore all [-s <startDate>] [-e <endDate>]
```

Parameters and Options:

Syntax	Required	Description
[-s <startDate>]	N	The beginning date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage.
[-e <endDate>]	N	The ending date range in which to specify the archived data. See Specifying Date Ranges for information of acceptable usage

Example:

```
$ everserve restore all -s 09/09/2002 -e 09/17/2002
```

Restoring Purged Records

The ***restore purged*** command retrieves a purged file to archive tables.

Command Syntax:

```
everserve restore purged {-d <directory>}
```

Parameters and Options:

Syntax	Required	Description
{-d <directory>}	Y	The full path and directory where the purged file resides.

Example:

```
$ everserve restore purge -d c:\myPurgedRecords
```

Error Recovery

It is possible that executing an archive, purge, or restore command might fail before completion of the operation. There are potentially many reasons why these operations might prematurely fail; for example shutting down either the Everserve server or database in the middle of the operation, would result in a failed operation.

In such situations, Everserve ensures that the data is not lost as long as the command is repeated. Whenever an archive, purge, or restore command fails, the command should be re-executed to bring the system to a valid state and to prevent loss of referential integrity.

If a failure occurs but the command is not reissued, the next time any one of the archive, purge, or restore commands is issued, Everserve will complete the previous incomplete task before proceeding with the current one.

Backing-up and Restoring Databases

Everserve provides an import/export utility that allows the backup and restore of the Everserve database to a secondary storage location. The backed-up data is machine independent allowing restoration of data to a different database server. There are two prerequisites for restoring an Everserve database:

- The Everserve database already exists.
- There is version compatibility between the Everserve database records to be restored and the backed-up data.

The following table lists the input expected by the utility.

Input	Description
Database Type	The type of database. Currently allows for MySQL, MS SQLServer, or a generic database. Database Type works with any database with a valid JDBC driver and a compatible Everserve schema defined.
Action	Possible values are backup or restore.
Directory	The directory that will house the backed-up data, or from which data is to be restored.
JDBC driver	The JDBC driver used to connect to the database.
JDBC url	The URL for the connection.
DB User	The DB username for the connection.
DB Password	The DB password for the connection.

For MySQL and MS SQLServer, the default JDBC driver and JDBC URL are set to the values currently supported by Everserve for these two engines. However, these values may be modified. All Everserve

tables, including archive tables, are backed-up to the directory specified. The archive tables will however be prefixed with the "archived_" tag.

The import/export command is used to either backup an Everserve database (using the export command that writes the database tables to a specified location), or to restore an Everserve database to a previous state (import from backed-up database tables).

To Back-up or Restore an Everserve Database:

1. Make sure the Everserve database is running.
2. Stop the Everserve service.
3. Open a command window and go to the C:\Program Files\Synchron Networks\Everserve\lib directory.
4. Enter either the **export** command or **import** command as shown below. Export is used to create a back-up of the database tables, import is used to extract backed-up database tables into the Everserve database.

```
everserve export
```

or

```
everserve import
```

5. Follow the instructions on the screen when prompted to run the import/export utility:

```
Enter database type:
```

```
Enter Enter action[import/export]:
```

```
Enter directory to backup to:
```

```
Enter Enter JDBC driver[org.gjt.mm.mysql.Driver]:
```

```
Enter Enter JDBC url[jdbc:mysql://localhost/everserve]:
```

```
Enter Enter DB User[everserve]:
```

```
Enter Enter DB Password[everserve]:
```

The system displays the values specified to run this utility:

```
DB type      : mysql
Action       : export
Directory    : tmp
JDBC driver  : org.gjt.mm.mysql.Driver
JDBC url     : jdbc:mysql://localhost/everserve
DB username  : everserve
DB password  : everserve
Enter Proceed[y]:
```

An acknowledgment of the operation will be displayed upon completion.

6. Restart Everserve.

System Logs

Everserve uses two separate logging features that enable System Administrators to monitor and audit system activity and processes. These logs include a server log maintained on each peer's local file system, and package and community information stored in the community manager's, publisher's, and relay's database. This section describes Everserve's logging mechanisms, provides information on how to view system logs and control logging verbosity, and describes the routine maintenance required to maintain Everserve system logs.

The topics in this section include:

- [Server Logs](#)
- [Process Logs](#)

Server Logs

Everserve's server log files are used primarily for activity monitoring and auditing. Each server log is maintained on each peers local file system and contains output from the Java Virtual Machine (JVM) running behind the scenes,

All Everserve systems maintain their own server log files on their local file systems. The main purpose for these server log files is to aid in diagnosing server problems. All log files (described in the following table) are located in the `\Synchron Networks\Everserve\server` subdirectory.

Log File	Description
<code>everserve.log</code>	Log file containing received output that corresponds to "stdout," which is considered "normal" output (such as messages). Each time Everserve is started on a system, a new <i>everserve.log</i> file is created.
<code>everserve_log.bak</code>	The contents of the old <i>everserve.log</i> file are appended to <i>everserve_log.bak</i> (the system creates this file if it does not already exist). After appending the old contents of <i>everserve.log</i> , Everserve writes its current logging information to this file when the system is restarted.
<code>everserve.err</code>	An error log file used to track error output such as stack traces from exceptions.
<code>everserve_start.log</code>	Log file showing timestamp when the device started or stopped an Everserve service.
<code>everserve_start.err</code>	An error log file that shows Everserve service shutdown and startup errors.

Auditing the System

Messages transmitted over the JMS messaging layer contain information pertinent to that device. For example, community managers can view all attempts to join the community; publishers can view the delivery status of all packages; relays can view a history of all packages received and passed to targets, and targets can view a history of all packages received.

Community manager and publisher devices have privileges and resources needed to view all commands issued to the system and can be accessed through the Web interface or command line interface.

Note: Although relays and targets do not have access to the Web interface, these devices can access this information using the command line interface. For information on how relays and targets can view process and messaging information, see the section [Information Services Commands](#) in this guide.

All asynchronous jobs that persist through server restarts are recorded in a jobs table. Examples of asynchronous jobs include activities such as joining a community, delivering a package, adding a peer to a community, or other messaging tasks where communication is generated from one device and sent to another, then after receiving the communication, a response is returned to the message originator (publisher).

Database Tables

Events are logged and recorded in the following database tables, with each containing details related to the corresponding event:

Log Table	Description
process	Lists each command issued that starts a process (either through the command line or through the GUI).
activity	Contains descriptions of parts of a process. Each process contains one or more activities, which is stored in the activity table.
receipt	Lists each time a publisher receives a receipt. The return receipt is logged in the receipt table.
receipt_element	Lists each element in the return receipt. Elements include results of applying file operations, standard output, standard error, and return code of all scripts that are executed as a result of opening a package.
package	Lists each package delivery command sent from the publisher.
package_element	Lists each element contained in the package. Package elements include the DTD file associated with the package, file-specs, directory-specs, command-specs, and scripts to run.
package_transport	Lists the transport used to deliver the package.

Viewing Server Log Files

By default, all log files are stored on each device's file system in the `\Synchron Networks\Everserve\server` subdirectory and can be viewed using an ASCII editor. Because log files are primarily used to troubleshoot server or device errors, performing case-sensitive keyword searches such as "Error" or "Exception" can be useful to track down specific problems.

Controlling everserve.log Verbosity

The data written to the `everserve.log` file is defined by the configuration file, `log4j.xml`, and can be modified as needed. If you are experiencing errors on an Everserve system it may be useful to increase the amount of data that is being recorded to this log file. By default, many entries in the logger section are commented out to decrease the amount of data being recorded to `everserve.log`. You can remove the comment notation (comments are found between the characters "`<!-- -->`") and begin

recording the specified data to the log file. The logging properties previously enclosed by the comment become active once the file has been saved with the changes.

Log4j.xml Properties

The following section describes how to change the logging properties to log more data than the default settings permit. Conversely, using the comment delimiters, you can exclude information from being logged to *everserve.log* as well.

The following example shows the different categories of data that can be recorded to *everserve.log*. The *log4j.xml* configuration file is located in the `\Everserve\server\config` subdirectory.

You can choose to include additional categories, or exclude categories of data that, by default, are turned on and logged to *everserve.log*. For tips on how to use the *log4j.xml* properties to troubleshoot system errors, see Synchron's Technical Support Web site at <http://support.synchronnetworks.com>.

Note: It is recommended that you create a backup copy of log4j.xml before editing this configuration file. This will be useful for future reference, and if needed, to revert back to the file's default settings.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration (View Source for full doctype...)>
- <!--
  Log4J configuration file
-->
- <log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"
  disable="null" disableOverride="null" debug="null" configDe-
  bug="null">
- <!--
=====
  Here is how you specify the format of and destination
  for all log entries written to STDOUT.  By changing the "value"
  in the "param" tag below, you change the format of ALL
  Everserve log messages. For details on the formatting
  strings that can be defined for "value", see "Log4J Configuration
  Quick Reference" at the end of this file.
-->
- <appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
- <layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d %-5p [%t] - %m\r\n" />
</layout>
</appender>
```

```

- <!--
=====
This entry defines the logging behavior for the all the
code written by Synchron Networks. The "priority value" tag
is used to set the logging level. The logging levels are:
    "fatal" - enables Asserts in all Synchron Networks code
    "error" - enables only error level messages (few)
    "warn" - enables only warning level messages (more, still few)
    "info" - enables info level logging (nearly none)
    "debug" - enables debug logging for entire Everserve product,
               all classes. This is the most verbose setting and
               will show debug tracing for every single class that
               has been enabled for log4j.

If this section is commented out, it will effectively disable
all Asserts in Everserve (which use the "fatal" level). By
uncommenting any of the categories described below, you can
selectively enable logging for specific areas of Everserve.
-->
- <category name="com.synchronnetworks" additivity="false">
<priority value="warn" />
<appender-ref ref="STDOUT" />
</category>
- <!--
=====
Here is how you specify the format of and destination
for all log entries written to STDERR. By changing the "value"
in the "param" tag below, you change the format of ALL
Everserve log messages. For details on the formatting
strings that can be defined for "value", see "Log4j Configuration Quick Reference" at the end of this file.

<appender name="STDERR" class="org.apache.log4j.ConsoleAppender" >
    <param name="target" value="System.err"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"

```

```

        value="%d %-5p [%t] - %m\r\n"/>
    </layout>
</appender>

-->
- <!--
=====
    Enable this category of logging to further investigate a Java
    exception that Everserve is throwing. Note that not all
    exceptions are fatal because Everserve has built-in recovery
    code to handle many exceptions.
-->
- <category name="com.synchronnetworks.util.EverOnException" additiv-
-   ity="false">
<priority value="error" />
<appender-ref ref="STDOUT" />
</category>
- <!--
=====
    Enable this category of logging to investigate why the Everserve
    server does not start or why it seemed to stop unexpectedly.

    Move the end-of-comment line up to the end of the next line
    to enable it.
ENABLED

    <category name="com.synchronnetworks.nile"   additivity="false">
        <priority value="debug" />
        <appender-ref ref="STDOUT" />
    </category>
end-of-comment
-->
- <!--
=====
    Enable this category of logging to investigate difficulties
    in one Everserve server connecting to another. If you need

```


further information, enable "com.synchronnetworks.messenger" described below.

Move the end-of-comment line up to the end of the next line to enable it.

ENABLED

```
<category name="com.synchronnetworks.roles" additivity="false">
  <priority value="debug" />
  <appender-ref ref="STDOUT" />
</category>
```

end-of-comment

-->

- <!--

=====

If you need further information after enabling "com.synchronnetworks.roles" (described above), enable this category of logging. This area is quite verbose so use with care when you are doing deliveries.

Move the end-of-comment line up to the end of the next line to enable it.

ENABLED

```
<category name="com.synchronnetworks.messenger" additivity="false">
  <priority value="debug" />
  <appender-ref ref="STDOUT" />
</category>
```

end-of-comment

-->

</log4j:configuration>

Log4j Configuration Quick Reference

The following table lists the *log4j.xml* configuration settings and provides description of their usage in this file:

Logging Parameters:

Parameter Tag	Value
%c	Category of the logging event
%C	Fully qualified class name of the caller
%d	Date of the logging event (example: %d{HH:mm:ss,SSS})
%F	File name where the logging request was issued (caution: extremely slow)
%l	Location information of the caller (caution: extremely slow)
%L	Line number from where the logging request was issued (caution: extremely slow)
%m	Application-supplied message
%M	Method name from where the logging request was issued (caution: extremely slow)
%n	Line separator
%p	Priority of the logging event
%r	Number of milliseconds since the start of the application
%t	Name of the thread that generated the logging event
%x	Nested diagnostic context associated with the thread
%%	A single percent sign

Format Modifier Examples:

The following table lists the modifiers used to format the output that *log4j.xml* generates.

Modifier	Output
%20c	Left pad with spaces if category is less than 20 characters long
%-20c	Right pad with spaces if category is less than 20 characters long
%.30c	Truncate from the beginning if category is more than 30 chars long
%20.30c	Left pad 20 chars + truncate from beginning if more than 30 chars
%-20.30c	Right pad 20 chars + truncate from beginning if more than 30 chars

everserve_log.bak File Maintenance

Each time Everserve is restarted a new server log is created and the old server log is appended to *everserve_log.bak*. *The growth of this file is dependant on the frequency in which server logs are created. In many cases, paring-down the everserve_log.bak file is required to handle log file growth. This file requires periodic size reduction maintenance to ensure the size of this backup log file does not become too large or cumbersome to work with, or that disk space to support this file becomes an issue.*

To reduce the size of *everserve_log.bak*, open the file with a text editor and either transfer older information to a separate file, or delete the unneeded information from *everserve_log.bak*.

Note: In cases where an Everserve system has been running continuously for an extended time, it may be necessary to pare-down the everserve.log file to control log file growth in the same manner as described above.

Process Logs

Process logs contain information about the processes that have run on an Everserve system. This information encompasses package deliveries and community configurations, and includes a log that lists all processing activity for the device. All information is recorded on an on-going basis and can be viewed from the command line using Everserve “show processes” command. In the case of community managers and publishers, process logs can be displayed from the Web interface (see the section [Viewing the Processes Log File](#)).

Appendix A:

Summary of CLI

Commands

The commands listed in this appendix summarize the syntax used to manage communities, deploy and monitor package delivery, and to make inquiries about system activities for your Everserve community.

The following table lists all Everserve commands and their parameters and options, and lists which role-type has privileges to execute the command.

Issuer	Command	Parameters
CM	add peer	{-p <peerName>} {-c <communityName>} [-r <role>] [-s <sender...>]
CM	add sender	{-p <peerName>} {-c <communityName>} {-s <sender...>}
PUB/CM	archive all	[-s <startDate>] [-e <endDate>] [-x]
PUB	archive deliveries	[-s <startDate>] [-e <endDate>] [-x] [-i <ID>]
PUB	archive deliverylog	[-s <startDate>] [-e <endDate>]
PUB/CM	archive processes	[-s <startDate>] [-e <endDate>]
CM	change community	{-c <communityName>} [-n <newCommunityName>] [-d <newDescription>]
CM	change peer	{-p <peerName>} [-n <newPeerName>] [-d <newDescription>]
CM	change peer	{-p <peerName>} {-c <communityName>} [-r <role>] [-s <sender...>]
CM	create community	{-c <communityName>} {-p <cmPeer>} [-d <description>]
CM	create peer	{-p <peerName>} {-h <hostname>} [-d <description>]
CM	create seed	{-c <communityName>}
CM	delete community	{-c <communityName>}

Issuer	Command	Parameters
CM	delete peer	{-p <peerName>}
PUB	deliver	[immediate] {-f <packageFileName>} [-c <communityName>] [-p <publisherPeerName>] [-i <ID>] [-l <peerList>]
ALL	env	
ALL	exit	
ALL	help	
ALL	join	{-c <communityName>}
ALL	list roles	
ALL	listxml	{-c <communityName>} [-p <peerName>]
ALL	pause	
PUB/CM	purge all	[-s <startDate>] [-e <endDate>] [-d <directory>]
PUB	purge deliveries	[-s <startDate>] [-e <endDate>] [-d <directory>]
PUB	purge deliverylog	[-s <startDate>] [-e <endDate>] [-d <directory>]
PUB/CM	purge processes	[-s <startDate>] [-e <endDate>]
ALL	quit	
CM	remove peer	{-p <peerName>} {-c <communityName>}
CM	remove sender	{-p <peerName>} {-c <communityName>} {-s <sender...>}
PUB/CM	restore all	[-s <startDate>] [-e <endDate>]
PUB	restore deliveries	[-s <startDate>] [-e <endDate>] [-i <ID>]
PUB	restore deliverylog	[-s <startDate>] [-e <endDate>]
PUB/CM	restore processes	[-s <startDate>] [-e <endDate>]
PUB/CM	restore purged	{-d <directory>}
ALL	resume	
ALL	run	{-f <fileName>}
ALL	show communities	[-c <communityName>] [-p <peerName>]
PUB	show deliveries	[-a] [-c <communityName>] [-p <peerName>] [-i <ID>] [-f <filter>] [-s <startDate>] [-e <endDate>]
ALL	show deliverylog	[-a] [-c <communityName>] [-p <peerName>] [-i <ID>] [-s <startDate>] [-e <endDate>]
ALL	show peers	[-c <communityName>] [-p <peerName>] [-h <hostName>] [-r <role>]
ALL	show processes	[-v] [-a] [-f <find>] [-s <startDate>] [-e <endDate>]

Issuer	Command	Parameters
PUB	show receipts	[-a] [-v] [-c <communityName>] [-p <peerName>] [-i <ID>] [-f <filter>] [-s <startDate>] [-e <endDate>]
ALL	show senders	[-c <communityName>] [-p <peerName>]
ALL	stop	
ALL	version	

Appendix B: Sample Script for Creating Communities

The commands listed in the sample script in this appendix were used to create the “Example” community and deploy and monitor package deliveries to the “Example” community. The steps performed using this scripts are illustrated in the section [Creating a Community](#).

```
# This file can be executed using the following command on a device that
# will be
#
# the Main Community Manager
#
# everserve run -f EverserveExample.EV
#
# Prior to executing this file, replace all example hostname values with
# real
#
# hostname values. If the hostname is not detected on the network, a
# warning will
#
# be issued.
#
# create the community
#
create community -c Example -p MainCM -d "Example Community"
#
```

```
# create and add the backup CM
#
create peer -p BackupCM -h BackupCMhostname -d "Backup Community Manager"
add peer -p BackupCM -c Example -r CM
#
# Create and add the publishers
#
create peer -p HRPublisher -h HRPublisherhostname -d "HR Publisher"
add peer -p HRPublisher -c Example -r Publisher
create peer -p OpsPublisher -h OpsPublisherhostname -d "Operations Publisher"
add peer -p OpsPublisher -c Example -r Publisher
create peer -p SalesRelay -h SalesRelayhostname -d "Sales Main Relay"
add peer -p SalesRelay -c Example -r Relay -s HRPublisher OpsPublisher
#
# Create and add the Relays
#
create peer -p ManufRelay -h ManufRelayhostname -d "Manufacturing Main Relay"
add peer -p ManufRelay -c Example -r Relay -s HRPublisher OpsPublisher
create peer -p RDRelay -h RDRelayhostname -d "R&D Main Relay"
add peer -p RDRelay -c Example -r Relay -s HRPublisher OpsPublisher
create peer -p USSalesRelay -h USSalesRelayhostname -d "US Sales Relay"
add peer -p USSalesRelay -c Example -r Relay -s SalesRelay
create peer -p AsiaSalesRelay -h AsiaSalesRelayhostname -d "Asia Sales Relay"
add peer -p AsiaSalesRelay -c Example -r Relay -s SalesRelay
```

```
#

# Show the senders for all Relays

#

show senders -c Example

#

# Create and add the targets

#

create peer -p HQTarget -h HQTargethostname -d "HQ Target"

add peer -p HQTarget -c Example -r Target -s SalesRelay

create peer -p SFTarget -h SFTargethostname -d "SF Sales Office Target"

add peer -p SFTarget -c Example -s USSalesRelay AsiaSalesRelay

create peer -p BostonTarget -h BostonTargethostname -d "Boston Sales
Office Target"

add peer -p BostonTarget -c Example -s USSalesRelay AsiaSalesRelay

create peer -p TaiwanTarget -h TaiwanTargethostname -d "Taiwan Sales
Office Target"

add peer -p TaiwanTarget -c Example -s USSalesRelay AsiaSalesRelay

create peer -p TokyoTarget -h TokyoTargethostname -d "Tokyo Sales Office
Target"

add peer -p TokyoTarget -c Example -s USSalesRelay AsiaSalesRelay

create peer -p LATarget -h LATargethostname -d "LA Manufacturing Target"

add peer -p LATarget -c Example -s ManufRelay

create peer -p DetroitTarget -h DetroitTargethostname -d "Detroit Manu-
facturing Target"

add peer -p DetroitTarget -c Example -s ManufRelay

create peer -p RaleighTarget -h RaleighTargethostname -d "Raleigh R&D
Office Target"

add peer -p RaleighTarget -c Example -s RDRelay
```

```
create peer -p AustinTarget -h AustinTargethostname -d "Austin R&D
  Office Target"

add peer -p AustinTarget -c Example -s RDRelay

#

# Show all peers in the community

#

show peers -c Example

#

# Create communities for Publisher backups

#

create community -c HRPubBackup -p MainCM -d "Community for HRPublisher
  Backup"

create community -c OpsPubBackup -p MainCM -d "Community for OpsPub-
  lisher Backup"

#

# show all of the communities

#

show communities

#

# add backup CM to backup communities

#

add peer -p BackupCM -c HRPubBackup -r CM

add peer -p BackupCM -c OpsPubBackup -r CM

#

# add publishers to backup communities

#

add peer -p HRPublisher -c HRPubBackup -r Publisher
```

```
add peer -p OpsPublisher -c OpsPubBackup -r Publisher

#

# create target peers to be used for backups and add them to their corre-
# sponding communities

#

create peer -p HRBackupTarget -h OpsPublisherhostname -d "Backup Target
for HR Publisher on Ops Publisher"

create peer -p OpsBackupTarget -h HRPublisherhostname -d "Backup Target
for Ops Publisher on HR Publisher"

add peer -p HRBackupTarget -c HRPubBackup

add peer -p OpsBackupTarget -c OpsPubBackup

#

# Show all peers in each community

#

show peers -c HRPubBackup

show peers -c OpsPubBackup
```


Appendix C:Third Party Software

Use of the software products listed below is hereby acknowledged.

FIORANOMQ

Copyright © 2002 Fiorano Software, Inc., All rights reserved

Bouncy Castle Crypto APIs

Copyright © 2000 The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>)

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

TWFreeTDS

Copyright © 2001 ThinWEB Technologies Inc.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. Licensor will provide a Warranty ONLY to those users who separately purchase a support package from Licensor that includes a Warranty. A support package can only be purchased by visiting Licensor's Web site.

Castor

Copyright 2000 © Intalio Inc. All Rights Reserved.

Developed by the ExoLab Project (<http://www.exolab.org/>).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Jakarta-Struts, log4j, Servletapi - 3.2.2, Tomcat-3.2, Xerces-J 1.3.1

Copyright © 2000 The Apache Software Foundation. All rights reserved.

The Apache Software License, Version 1.1

Copyright © 2000 The Apache Software Foundation. All rights * reserved.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>).

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Glossary

The following terms are used in this document:

Term	Definition
Community	A community is a set of peers, together with the role that each peer has within that community, and routing information that specifies how packages are routed to each within that community.
Community Manager	The community manager is a device installed with capabilities to create communities and peers, and defines the routing information and topology used in the Everserve community.
Deployment	The process of propagating application files to target systems, then automatically installing the application without human intervention. If a target is unavailable to receive the package, the package is queued until delivery can be made.
Everserve Device	A physical computer system that has Everserve software installed on it.
Everserve Scripts	An Everserve script is a script that will be executed on a set of Everserve targets. Everserve scripts may be intrinsic scripts or shell scripts. An intrinsic script is executed directly by the Everserve software in the most reliable means possible (such as “resume”), whereas a shell script is executed as a shell command (such as “less -l” or “dir”). Intrinsic scripts work on all Everserve computer systems, regardless of the operating system and operating environment, whereas shell scripts are executed by the shell environment of the target system. Thus, shell scripts may be operating system dependant. Everserve makes no attempt or guarantee to execute shell scripts in an operating system independent manner.
File Operation	A file operation is an operation to be performed on a file system object, which includes operations such as: create directory, create link, create file, rename file, update file, and so on. A file operation includes the data necessary to carry out the operation, such as the file name and the file contents. In the particular case of a file update, only the data that actually changed is required to be part of the file operation; this allows for an efficient mechanism for delivering small changes to large files. Publishers create sets of file operations, insert them in packages, and send packages to targets.
File set	A file set a set of file system objects, including: directories, files, and links (both hard and soft).

Term	Definition
First Level Relay	A first level relay is a relay peer that is connected to a publisher. If the community uses multiple levels of relays, a first level relay will receive packages from a publisher, then pass the package on to other relays in the community.
Opening a Package	Opening a package is the process of receiving a package from a queue, verifying the digital signature of the package, decrypting the package using the package encryption key, executing the pre-script (if any), applying the set of file operations (if any), executing the post-script (if any), and finally sending the package return receipt back to the publisher.
Package Delivery	Package delivery is the process of sending a package from a publisher to a set of targets.
Package	A set of files and scripts that are delivered to remote computers. All packages are digitally signed by the publisher that originates package delivery. All files and scripts to be delivered are defined by the package specification.
Package Specification	A package specification is an XML file that contains the list of files, scripts, and commands that define the operations to be performed on targets.
Package Update	A package update is an update to a previous package delivery, intended to bring the previous, old state of the package on remote systems up to date. Everserve delivers package updates in an optimal manner, sending only the delta changes to update the peers.
Peer	A device (for example, a personal computer or server) that is a member of a community having one of the following roles: community manager, publisher, relay, or target. If a peer belongs to more than one community, it may have a different role in each community.
Peer Roles	A peer may have one role within the context of a particular community. The following peer roles are supported: community manager, publisher, relay, and target.
Publisher	A publisher is a peer that defines the contents of a package and initiates package delivery to a set of targets. After targets open and execute the package, the results are sent back to the publisher in the form of a return receipt. Return receipts are stored and accessed in the publisher's database.
Receiver	A receiver is a peer that accepts packages from a community sender. Receivers can be relays or targets.

Term	Definition
Relay	A relay is a peer that receives a package from either a publisher or another relay and sends the package to targets. Return receipts from the targets are sent to the sending relay, then forwarded to the originating publisher.
Replication	The process of propagating files between Everserve peer systems. If a peer is unavailable to receive the package (replicated file), the package is stored in queue until a delivery can be made.
Return Receipts	A return receipt is the result of opening a package. Included in the return receipt is the standard output, standard error, and return code of all scripts that were executed. It also includes the results of applying the file operations. Return receipts are sent by targets to the publishers from whom the package originated.
Seed File	A file in zip format created by the community manager in the <code>\Synchron Networks\Everserve\server</code> directory following execution of the "create community" command. This file contains all the information needed to enable an Everserve device to receive packages
Sender	A sender is any peer in the community that either initiates or passes a delivery to another peer. Senders can be publishers, which initiate a delivery, or relays, that pass a delivery on to the connected targets. Community managers and targets can not be senders.
Target	A target is a peer that receives a package. When a target receives a package, it opens the package, executes the commands or scripts contained in the package, and sends a return receipt back to the originating publisher.
Transport	The means by which Everserve systems communicate with each other.

Index

A

- add peer command 116
- adding
 - multiple peers to a community 122
 - senders 118
- administrative commands
 - add peer 116
 - add sender 118
 - change peer 124
 - change sender 126
 - create community 110
 - create peer 115
 - create seed 113
 - delete community 112
 - delete peer 130
 - remove peer 129
 - remove sender 126
- archiving all records 237
- archiving deliveries 234
- archiving deliverylog 236
- archiving processes 235
- archiving records 233
 - best practices 233
 - specifying date ranges 233

C

- change peer 124
- change sender 126
- command line interface
 - add peer 116
 - add sender 118
 - change peer 124
 - change sender 126
 - create community 110
 - create peer 115
 - create seed 113
 - delete community 112
 - delete peer 130
 - deliver 131
 - interactive shell 106
 - pause 108
 - remove peer 129

- remove sender 126
- resume 108
- run 133
- commands
 - change community 111
 - general administrative 107
 - list 147
 - listxml 147
- communities
 - adding multiple peers to 122
 - changing definition of 189
 - changing definitions of 111
 - complex community, example of 150
 - creating 110
 - creating with Everweb 186
 - creating, example of 112
 - deleting 112
 - deleting definition of 190
 - descriptions 109
 - managing 109
 - managing with Everweb 186
 - removing peers 200
 - using multiple community managers 151
 - using multiple publishers 151
 - using tiered relays 151
- community manager
 - system requirements 30
- configuration
 - Everserve, for Internet traffic 76
- connection rules
 - default connections 119
 - specifying connections 120
 - when adding peers 119
- conventions
 - used in document 16
- create community 110
- create peer 115
- create seed 113
- creating
 - communities using Everweb 186

D

- database
 - backing-up 248
 - installing MySQL 32
 - installing MySQL on Solaris devices 34

- installing MySQL on Windows devices 32
- optimizing 31
- requirements 29
- security 30
- setting up 30
- date ranges
 - for archiving records 233
 - used with list and show commands 135
- delete
 - peer definitions 204
- delete community 112
- delete peer 130
- deliver command 131
- deliveries
 - archiving records of 234
 - purging records of 239
 - restoring records of 243
- deliverylog
 - archiving records of 236
 - purging records of 240
 - restoring records of 234, 235, 236, 239, 240, 241, 243, 244
- devices
 - activating 128
 - joining a community 128
 - obtaining a list of 23

E

- env settings
 - using the CLI 147
- Everserve
 - add peer command 116
 - change community command 111
 - change peer command 124
 - controlling operative states 107
 - create community command 110
 - create peer command 115, 130
 - create seed command 113
 - deliver command 131, 133
 - Internet set-up configuration 76
 - introduction to 13
 - list commands 147
 - pausing and resuming 210
 - remove peer command 129
 - server logs 252
 - viewing log files 206, 207
- Everweb

- changing community
 - definitions 189
- changing peer definitions 199
- connection to 182
- creating communities 186
- creating peers 191
- deleting community
 - definitions 190
- deleting peer definitions 204
- logging off 211
- managing communities 186
- managing peers 191
- page navigation 185
- removing a peer 200
- server operations 210
- status indicators 185
- viewing peer definitions 205
- viewing process log files 207
- viewing server log files 206
- viewing system logs 206
- examples
 - add peer 117
 - add sender 118
 - change community
 - definition 111
 - change peer 124, 125
 - create community 111
 - create peer 115
 - create seed 113
 - creating a new community 110, 142
 - delete community 112
 - delete peer 130
 - delivering a package 132, 136, 138, 139, 143, 145, 234, 235, 236, 237, 239, 240, 241, 242, 243, 244, 245, 246, 247
 - pausing Everserve 108
 - remove peer 129
 - resuming Everserve 108
 - run 133

F

- failover
 - planning for 26

H

- help
 - CLI commands and syntax 148

I

- information commands
 - command line interface 135
 - specifying date ranges 135
- informational commands
 - listing community
 - definitions 147
- installation
 - interactive 36
 - JMS only on Solaris 83
 - JMS only on Windows 77
 - JMS server only 76
 - pre-install checklist 28
 - target only on Windows 36
- installations
 - non-interactive on Solaris 72
 - non-interactive on Windows 61
 - PKG for Solaris 64
 - post-install tasks 85
 - silent, Solaris 72
 - silent, Windows 61
 - Solaris 64
 - Windows custom, all roles 43
 - Windows registry entries 85
- interactive shell
 - invoking 106
- Internet
 - configuring Everserve for 76
- inventory
 - obtaining community device
 - list 23

J

- JMS
 - installing 76
 - Solaris installations 83
 - Windows installations 77
- join
 - becoming a community
 - member 128

L

- list and show commands 135
 - date range usage 135
- list commands 147
 - list env settings 147
 - list Everserve version 148

- list roles 147
- listxml 147
- list Everserve version
 - using the CLI 148
- list roles
 - using the CLI 147
- listing transports 147
- listxml
 - using the CLI 147
- log files
 - process log 207
 - server log 206
- log4j.xml
 - config reference 258
 - server log config file 253
- log4j.xml properties 254
- logical peer groups
 - planning for 24
- login
 - Everweb 182

M

- MySQL
 - installing 32
 - installing on Solaris devices 34
 - installing on Windows devices 32

O

- operative states
 - pause 108
 - resume 108
 - start command line interface 107
 - stop command line interface 107

P

- pausing Everserve 108
- peer definitions
 - changing using Everweb 199
- peers
 - activating 128
 - adding 116
 - adding multiple peers 122
 - changing 124
 - changing connections 126
 - changing, example of 125
 - connection rules 119
 - creating 115

- creating with Everweb 191
- creating, example of 115
- default connection rules 119
- deleting 130
- deleting definitions 204
- descriptions 114
- joining a community 128
- managing 114
- managing with Everweb 191
- removing 129
- removing connections 126
- removing from
 - communities 200
- removing, example of 127
- specifying connections 117, 120
- viewing definitions 205
- planning
 - pre-install checklist 28
- planning your community
 - designing topology 24
 - failover 26
 - logical groupings 24
 - redundancy 26
- process logs 259
- processes
 - archiving records of 235
 - purging records of 241
 - restoring records of 245
- properties file
 - sample of, Windows 62
- publisher
 - system requirements 30
- publishing commands
 - deliver 131
 - run 133
- purged records
 - restoring 247
- purging all records 242
- purging deliveries 239
- purging deliverylog 240
- purging processes 241
- purging records 238
 - best practices 238

R

- records
 - archive all 237
 - archive deliveries 234
 - archive deliverylog 236
 - archive processes 235
 - archiving 233

- purge all 242
- purge deliveries 239
- purge deliverylog 240
- purge processes 241
- purging 238
- restore all 246
- restore deliveries 243
- restore deliverylog 234, 235, 236, 239, 240, 241, 243, 244
- restore processes 245
- restore purges 247
- restoring 243
- redundancy
 - planning for 26
- redundant routing
 - example of 150
- relays
 - system requirements 30
 - using in communities 25
- remove sender 126
- removing peer 129
- response file
 - sample of, Solaris 73
- restoring
 - archived records 243
 - database 248
 - purged records 243
 - restoring all records 246
 - restoring deliveries 243
 - restoring deliverylog 234, 235, 236, 239, 240, 241, 243, 244
 - restoring processes 245
 - restoring purged records 247
- resume
 - Everserve command 108
- resuming Everserve 108
- run command 133

S

- seed file
 - recreating 113
- sender connections
 - specifying 117
- senders
 - adding 118
 - changing 126
 - changing for peers 126
 - default connections 119
 - peer connections 120
 - removing 126

- removing peers 126
- rules when connecting 119
- server log
 - everserve_log.bak 259
 - setting verbosity 253
 - viewing log files 253
- server logs 252
 - for system audits 252
- show commands
 - show communities 136
 - show deliveries 144
 - show deliverylog 145
 - show peers 137
 - show processes 140
 - show receipts 142
 - show senders 139
- show communities
 - using the CLI 136
- show deliveries
 - using the CLI 144
- show deliverylog
 - using the CLI 145
- show peers
 - using the CLI 137
- show precesses
 - using the CLI 140
- show receipts
 - using the CLI 142
- show senders
 - using the CLI 139
- silent installations
 - on Windows 61
 - sample, Solaris response file 73
 - sample, Windows properties file 62
- Solaris
 - installing Everserve 64
- start
 - starting Everserve 107
- status indicators
 - for Everweb pages 185
- stop
 - stopping Everserve 107
- system configuration
 - system requirements 29
- system logs
 - viewing 206
- system requirements 29, 30
 - database 29
 - supported platforms 29

T

- target installs
 - only, on Windows 36
- targets
 - system requirements 30
- technical support 15
- terminology
 - used in document 17
- topology
 - planning for communities 24
 - using relays 25