Everserve™

Product Overview

Part No. 2.01-1.8-OVR-03

Table of Contents

Introduction

This *Product Overview* provides a conceptual look at Synchron Network's secure application and content deployment software. Also in this guide you will find information on the methods and processes Everserve uses to securely and reliably deliver content and deploy applications to massive numbers of globally distributed devices.

The topics discussed in this Introduction include:

About this Guide

This *Product Overview* is designed to help you understand the principle concepts of how Everserve is used to deploy applications and propagate data across devices. This guide is an overview only—it provides a high-level view of Everserve's community configuration models and secure package distribution. In this *Product Overview* you will find:

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.

Overview of Everserve describes Everserve's deployment model and introduces deployment topology concepts to consider when configuring your communities of Everserve devices.

Community Management describes the process of creating a new *Everserve* community. This section contains a flow chart of the primary tasks performed by a System Administrator to create a community and deliver packages.

Publishing Packages describes many of the mechanisms *Everserve* uses to securely and reliably deliver packages to other *Everserve* devices in the community.

Where to Find Additional Information

This *Product Overview* provides high-level information on the processes used to create communities and deliver packages. Refer to the following user guides for additional information:

- *System Administrator's Guide* for information on how to install Everserve, create communities, and perform record and database basic maintenance.
- *Publisher's Guide* for information on how to create package specifications and deliver packages to communities.

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://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.
Bold Italic	Used to identify a ***new term***. 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.
Italics	Indicates variable values you must provide (for example, you may be prompted to supply the name of a *file* for *fileName*). 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 `start` command" means type *start* 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 the *System Administrator's 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 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.
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.

Overview of Everserve

The core feature of Everserve is its secure, reliable, and efficient delivery of ***packages*** to a potentially very large set of computers. A package may contain data, applications, upgrades, or files and scripts that are sent to remote computers.

This Overview provides a high level description of Everserve. The topics discussed in this Overview include:

- Using Everserve
- Key Features
- Deployment Process

Using Everserve

Everserve can be used to solve a variety of problems associated with distributing data securely, deploying applications, and ensuring the latest software versions, updates, and patches are maintained on all networked systems. The following sections describe the most common uses of Everserve. For additional examples and use cases, visit Synchron's Technical Support Web site at http://www.synchronnetworks.com/support.

Software Deployment: Installations, Updates and Patches

Deploying new software applications, upgrades, and patches is time consuming and often difficult to schedule without added downtime to network servers and desktops. In most enterprise environments, software deployment competes for IT cycles with many complex business projects (such as migrations, upgrades, architectural re-designs and back-end network rework) that are taking place. Because software deployment typically requires a manual process of visiting each desktop or server, enterprises typically hold off on deployment of upgrades to servers and desktops until there is a severe need. Installations, upgrades, and patches often are not deployed until the versions being used are deemed inadequate, which can lead to security risks and misconfigured systems.

Everserve securely delivers applications, content, and data to numerous servers and desktops in a single instance with little or no user intervention or disruption of service. The mass delivery of software upgrades, data and content assures the timely and secure deployment of critical patches and updates that end users rely upon to be productive in their day-to-day job functions.

Security Updates

Ensuring all desktop systems and network servers are protected from unauthorized access, attacks, and hacking has become the computing industry's primary focus. In most cases, responsibility to ensure the latest security updates and patches are installed on systems falls on each employee, rather than a central focus group such as an IT department. This often leads to many computing systems operating with old security software applications. Everserve can securely and efficiently deploy any security application, update, or patch to servers and desktop systems without the need for human intervention. Using Everserve in this fashion can eliminate many of the security risks that enterprises are faced with today, and places the security of the enterprise network back into the hands of IT professionals.

Secure Content Distribution

Today, enterprises and businesses are unable to securely and reliably distribute files of arbitrary content types or large size to devices outside their LAN. Typically, electronic distributions are used when files are small and low risk, while large files or sensitive data is generally distributed using CD's and physical delivery. This current method of content distribution is not conducive to large scale distribution, is labor intensive, can lead to security transgressions, and is prone to error.

Everserve can be used to securely distribute content to mass numbers of devices, regardless of file type or size. Everserve employs a unique method of persistence, ensuring deliveries to other Everserve devices are completed even if a network or Internet connection is lost temporarily.

Everserve combines an automated embedded PKI engine with rich features and functionality to enable secure exchange of confidential and proprietary information. This technology protects communications over the Internet despite the Internet's notoriously poor security. It also protects in-house file transfers so that unauthorized parties within an organization do not gain access to sensitive information.

Everserve's secure, efficient, and reliable electronic communication opens many opportunities for enterprises and businesses by improving content distribution processes while also reducing internal costs and eliminating manual processes.

Key Features

Everserve is recognized by enterprises, financial institutions, health care organizations, and telecommunication companies as a platform that securely and reliably deploys applications and data to massive numbers of devices through remote command execution. To ensure package delivery, Everserve handles connectivity interruptions between servers, storing packages in queue until delivery can be made. Everserve's key features include:

Content Distribution and Application Deployment	Everserve paves the way for massive secure deployment of applications and files to distributed devices across any network environment. Everserve's delivery and execution tools solve the problems surrounding software distribution to a large number of unattended, disparate and dispersed network devices by automating application deployment over any network, including the Internet. It ensures each device receives the proper file set, even if it is offline at the time of the delivery or if the delivery fails due to an interrupted transmission in the underlying network.
Security	Everserve employs Public Key Cryptographic Standards (PKCS #7) techniques and SSL standards to ensure secure transmissions of data and applications. It employs a unique double-layered challenge-response protocol for all **peers** wishing to join a community. Everserve ensures appropriate authentication and access for all peers in the community, ensuring that only intended recipients receive, open, and execute packages.
Replication	Everserve's propagation mechanisms are unidirectional in nature; that is, **packages** flow along a directed, a-cyclic network of computer systems. Everserve reports on the status of packages it delivers so that a System Administrator may determine whether the files and scripts have been delivered and executed properly.

Reliability

Everserve handles connectivity interruptions between computer systems under its control, storing packages for delivery when the connection is restored.

Messaging protocols and messages sent through the queuing system are defined using XML. This allows third-party systems to interact with Everserve and allows for the development of standards based on Synchron's technology. In addition, Everserve supports redundant **relays** to provide fail-over and dynamic, wide-area load balancing.

Scalability

A unique feature of Everserve is its ability to scale to thousands of servers and peers with minimal human intervention. This measure of scalability is accomplished by using community relays to simultaneously broadcast packages to a large number of computers at the same time.

Deployment Process

The core feature of Everserve is the secure, reliable, and efficient delivery of Everserve packages to a potentially very large set of computers. An Everserve package may be loosely defined as a set of files, commands, and scripts that are delivered to other Everserve systems. Everserve packages are used to:

- Deliver content and data.
- Execute commands, scripts, and batch files.
- Install, upgrade, and maintain applications.

Everserve scales to a very large number of computer systems allowing packages to be simultaneously broadcast to a large number of computers at the same time. Everserve compresses and encrypts all packages before transmission to ensure safe and efficient delivery of all packages to community peers.

All Everserve devices undergo authentication to become a trusted member of an Everserve community. As Everserve devices join a community, digital certificates and public and private keys are passed between the community manager and the device to establish a trust model. As packages are sent from the community's publisher to each Everserve device, Everserve encrypts the package content, thus enabling another level of security for all content and applications deployed to devices.

Everserve reports on the status of packages it delivers, so that a system administrator or third party software system may determine whether the files and scripts have been delivered and executed properly. Everserve automatically handles connectivity interruptions between computer systems under its control, storing packages for guaranteed delivery when the connection is restored.

Community Management

Community management encompasses the primary tasks a System Administrator performs to create and populate a new community using Everserve. Once a community is established, community management tasks typically include maintaining peer and community definitions, as well as routine record and database maintenance.

This section provides a high-level view of the steps to create and establish a community, and describes several common topologies you can employ for your Everserve communities. For in-depth information and systematic instructions on how to manage communities, refer to the *System Administrator's Guide*.

The topics discussed in this section include:

- Steps to Create a Community
- Community Topologies
- Activating a New Everserve Device

Steps to Create a Community

Before you begin using Everserve for application deployment, you must first define a community to include a community manager, one or more publishers, one or more relays (relays are optional), and targets. Figure 1 shows the typical tasks that an administrator might perform to set up and configure an Everserve community. Although package delivery is a function of the community publisher, this diagram shows the hand-off point in this process from the community manager to all peers, and to the publisher.

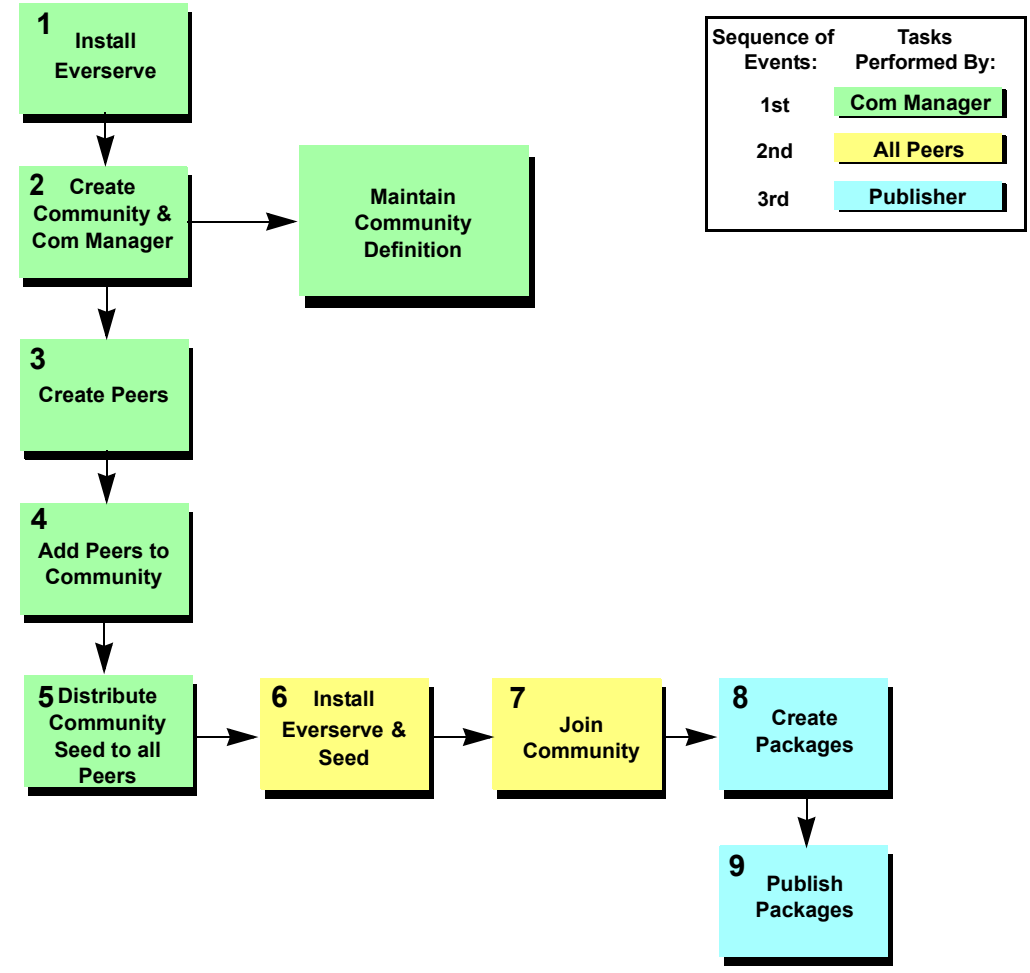

Figure 1 Typical Work and Process Flow

The following sections briefly describe each task shown in Figure 1.

Step 1 - Install Everserve

All peers that are to be included in an Everserve community must have Everserve installed and running. The installation can be completed any time prior to publishing a package. Any peer that does not have Everserve installed will not be entitled to receive packages from the community publisher. Refer to the section Installation in the *System Administrator's Guide* for information on installing Everserve.

Step 2 - Create the Community and Community Manager

Once Everserve is installed with community manager capabilities onto the device that will be used as the community manager, the System Administrator can then create an [empty] community, and a the community manager of the new community. After the community and community manager have been created, the community manager can create and add peers to the community (see Step 3 below). For information and instructions on how to create a community and community manager, refer to Creating a Community in the *System Administrator's Guide*.

Step 3 - Create Peers

From the community manager device, the System Administrator creates a peer definition for each peer that will be included in the community. Peer definitions include a unique name for the peer, a description for the peer and the hostname or IP address of the peer. See Managing Peers in the *System Administrator's Guide* for information and instructions on how to create peer definitions.

Step 4 - Add Peers to the Community

Once peer definitions have been created they can then be added to the community. From the community manager device, the System Administrator adds each peer to the community, assigning a role for the peer at that time. Peer roles can be any of the following:

Community Manager:	Has the right to modify the definition of a community by adding or deleting peers from a community, by changing peer roles, or routing information of the peers in the community. There must be at least one community manager for each community.
Publisher:	Creates package specifications and initiates package delivery to a set of targets. Each community must have at least one publisher.

Relay:	Receives a package from a publisher, then forwards the package to the intended target. If the relay is also a target, it executes the package before sending the package to recipient targets.
Target:	Receives and executes packages sent from the publisher or relay.

See Adding Peers to a Community in the *System Administrator's Guide* for additional information on how to populate the community with peer definitions and assign peer roles.

Step 5 - Distribute Community Seed to All Peers

Once peer definitions have been created and added to the community, the community definition is essentially complete. When a community is created, a community **seed file** that contains routing information is automatically generated. The community seed file is the entitlement for the peer to join the community to receive packages. The System Administrator must distribute the seed file to each peer in the community. See Activating a New Everserve Device in the *System Administrator's Guide* for additional information.

Step 6 - Install Seed on Each Peer

Before a peer is entitled to join a community it must obtain the community seed file generated by the community manager. This seed file can be distributed to each peer by copying it to distribution media (CD or floppy disk) and distributing it to the community peers, or by copying the seed file to an network location that is accessible to the peers.

Step 7 - Join the Community

To become an active member in the community, each peer must join the community. After the seed file is copied to the peer's local file system, the peer issues an Everserve command to join the community, which initiates a challenge-response protocol between the community manager and the peer. This protocol establishes a trust model for the peer to ensure its identity and role in the community. Once joined, a peer can begin to participate in the community as designated by its role (that is, a publisher can begin to send out packages, a relay can begin to receive and forward packages, and a target can begin to receive packages). See Joining the Community in the *System Administrator's Guide* for additional information.

Step 8 - Create Packages

Any peer designated as a community publisher is entitled to create and deliver packages. A publisher peer uses Everserve tools to create a package specification that defines the contents of a package. The package specification contains a list of files names, directories, applications, scripts, or any other digital data located on the publisher's device that the publisher wants to send to peers in the community. See Creating a Package Specification in the *Publisher's Guide* for additional information.

Step 9 - Deliver Packages

Once the contents of a package has been defined, the publisher sends the package to the community peers using a secure, reliable transport mechanism and protocol. See Delivering Packages in the *Publisher's Guide* for additional information.

Community Topologies

Content distribution and application deployment is the means of handling the distribution and execution of packages that contain applications, upgrades, patches, and/or data. There are several ways to set up your community topology to optimize distribution of content and deployment of applications and data to target devices. The following sections describe the fundamental topology concepts used with Everserve's distribution and deployment models.

One-to-One Deployment

The most basic model of content and application deployment is between two peers in a community. This is a one-to-one deployment configuration. In this model, once a new peer is added to a community, then joins the community, the publisher is then able to send package(s) to the new ***target***.

The publisher creates a package specification that identifies the package content, encrypts the package and its contents, and sends the package to the target peer. The target peer is restricted to receiving packages from the publisher only, propagating the package to its own file system. The target peer returns to the publisher output from the execution of the package specification to the publisher—it does not send packages to the publisher. Figure 2 illustrates a one-to-one deployment model between a publisher and a target.

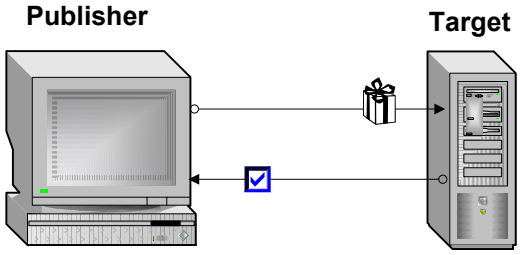

Figure 2 One-to-One Deployment Between Publisher and Target

In this example, the package is deployed as follows:

1. The publisher compresses the applications, data files, commands, and scripts into a package, then encrypts and digitally signs the package for transport.
2. The publisher sends the package to the target.
3. The target opens the package and applies the changes (install/upgrade applications, propagate files and directories, and so on) as specified by the package specification.
4. After executing the contents of the package, the target sends a ***return receipt*** back to the publisher.

Fan-Out Deployment

Figure 3 illustrates a fan-out approach to deploying applications and content. This is an expanded version of the one-to-one deployment model, where application and data files are packaged by the publisher and sent to all targets in the community.

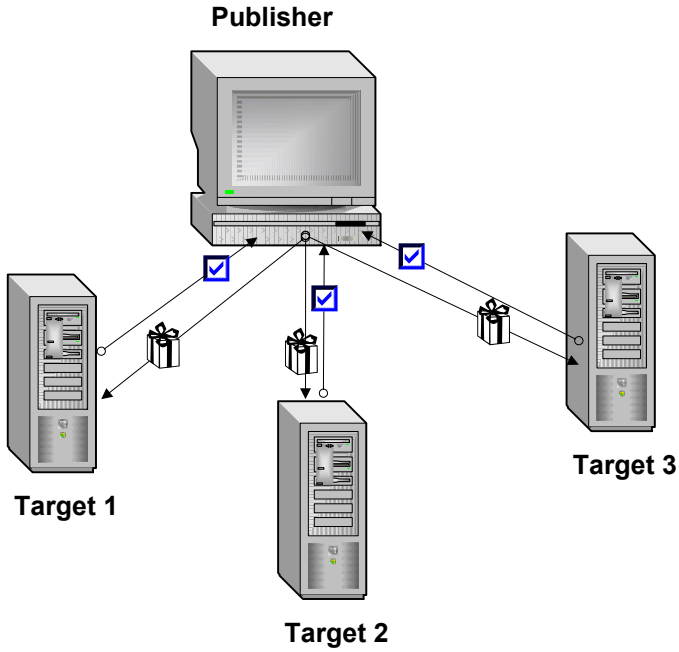

Figure 3 Fan-Out Deployment Between Publisher and Several Targets

In this example the package is deployed as follows:

1. The publisher compresses the applications, data files, commands, and scripts into a package, then encrypts and digitally signs the package for transport.
2. The publisher sends the package to all targets in the community, *Targets 1, 2, and 3.*
3. Each target opens the package and applies the applications and files as specified in the package specification.
4. After executing the package, each target (*1, 2, and 3*) sends a return receipt back to the publisher.

Relay Deployment

Relays offer a unique means of increasing scalability while improving network performance. A typical server-class relay (with four CPU's and abundant RAM) can effectively handle thousands of simultaneously connected targets.

Figure 4 illustrates a relay approach to the one-to-one deployment model, where application and data files are packaged by the publisher, sent to a relay, then ultimately sent to the recipient target.

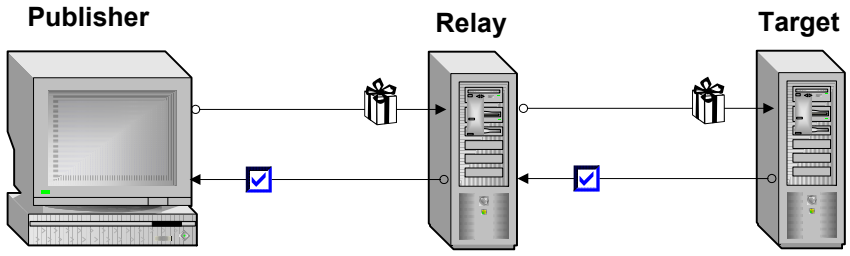

Figure 4 Relay Deployment

In this example the package is deployed as follows:

1. The publisher compresses the applications, data files, commands, and scripts into a package, then encrypts and digitally signs the package for transport.
2. The publisher sends the package to the relay.
3. The relay forwards the package to the intended target.
4. The target opens the package and applies the changes (install, upgrade, propagate files and directories, and so on) as specified in the package specification.
5. After executing the package, the target sends a return receipt back to the relay, who then forwards the return receipt to the publisher.

Combining Fan-Out and Relay Deployment

Figure 5 illustrates a combined approach to the fan-out and relay deployment models where application and data files are packaged by the publisher, sent to the community relays, and ultimately sent to all other targets in the community. Note that the publisher receives and logs return receipts from all targets receiving the package.

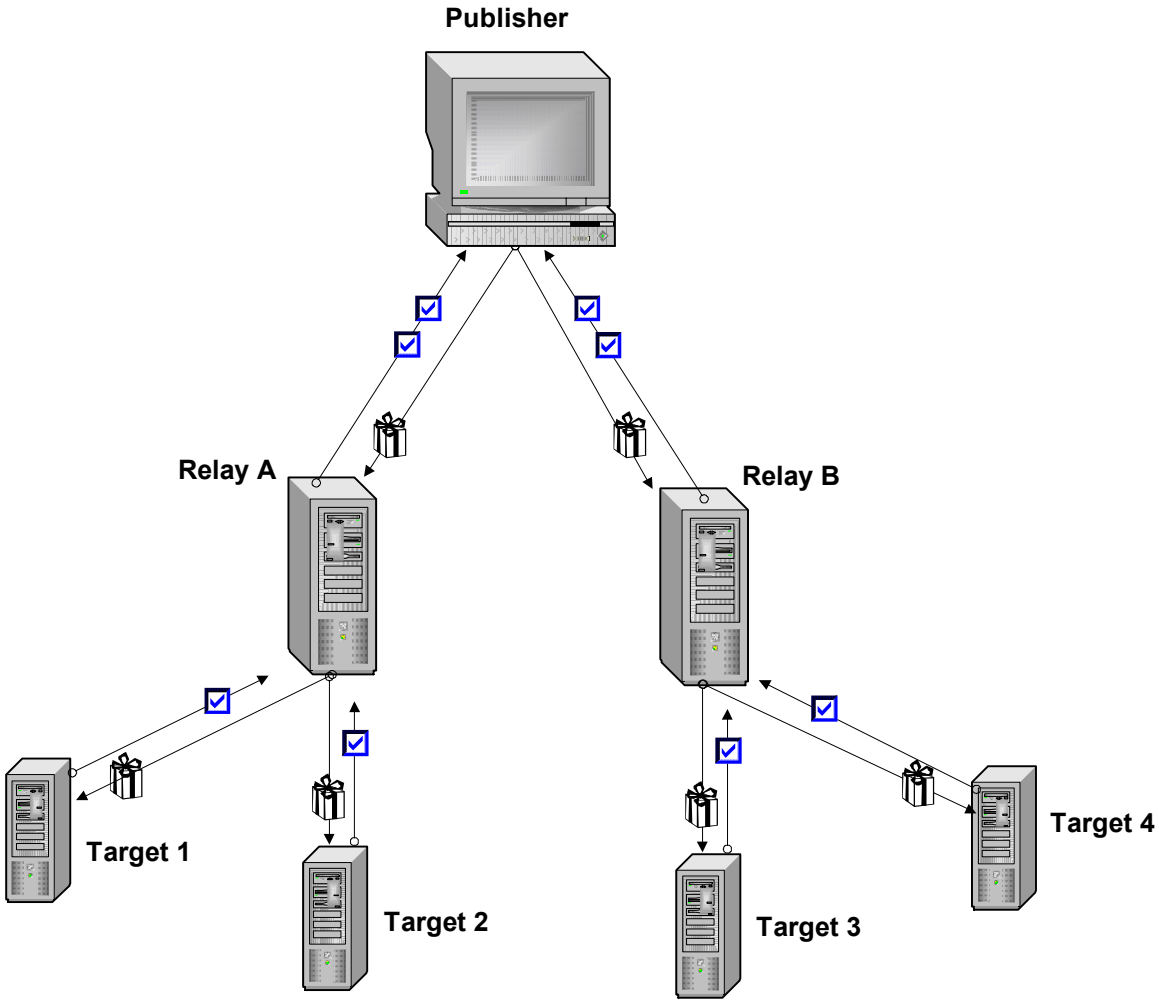

Figure 5 Combination Fan-Out and Relay Deployment

In this example the package is deployed as follows:

1. The publisher compresses the applications, data files, commands, and scripts into a package, then encrypts and digitally signs the package for transport.
2. The publisher sends the package to the community relays, *A* and *B*.
3. Relay *A* sends the package it received from the publisher to Targets *1* and *2*. Targets *1* and *2* process the package and send a return receipt back to *Relay A*. *Relay A* sends the return receipts to the publisher.
4. Simultaneous to Step 3, *Relay B* sends the package it received from the publisher to Targets *3* and *4*. Targets *3* and *4* process the package and send a return receipt back to *Relay B. Relay B* sends the return receipts to the publisher.

Activating a New Everserve Device

After the community and community manager have been created, a community ***seed file*** is automatically generated. This seed file is distributed to each Everserve device in the community. Each device then joins the community to become an active and trusted community member.

Using the information in the seed file, Everserve establishes a secure communication channel between the community manager and the new device. This secure channel allows the new device to validate the identity of the community manager by verifying the community manager's certificate. 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.

If the device is a target, it connects to the queue of the device that sends packages to community peers (either a publisher or a relay). The new device is now able to receive packages.

Publishing Packages

Publishing is a term used to describe the mechanisms and processes Everserve uses to securely, efficiently, and reliably transmit packages to community peers. Packages may contain content and data, applications, executable commands, or batch/script files intended for target execution. Packages can be sent to all peers in a community, or to a subset (or list) of peers. Each peer that is sent a package returns a receipt back to the publisher that originated the delivery.

The topics discussed in this section include:

- Package Specifications
- Package Delivery Process
- Delivery Mechanism
- Maintaining Persistent Data

Package Specifications

A package specification is an XML file that contains the list of files, applications, datasets, commands, scripts, and batch files to deliver to a target device. Package specifications are created to instruct a target to copy a dataset, install and application, or to execute scripts or commands. Flags can be set in the package specification to instruct the system to automatically scan certain directories and files on the publisher's system for changes, and if changes are found, to propagate the changes to the peers in the community when the package is resent.

When creating package specifications that include executable commands, it is important to construct packages that will be understood by the recipient target system. That is, a Windows target can receive a package containing scripts and commands that are UNIX specific, but it will not be able to apply the package to its file system as it does not understand the UNIX OS specific language. Conversely, packages containing Windows specific commands and scripts will not be understood by UNIX devices. Package specification templates are provided with Everserve and can be used to create custom package specifications.

Package Delivery Process

Once the package specification has been created, it is delivered to targets that have Everserve installed, are configured and entitled to receive packages, and are a member of an Everserve community. When a target receives a package, it opens the package and executes the scripts and commands contained in the package.

Delivering Packages to Everserve Peers

All package deliveries are initiated by a community publisher. The publisher can choose to deliver a package to all peers in a community, to a list of peers in a community, or to a single peer in a community. Once package delivery begins, Everserve bundles the package contents then sends the package to the peers. Each recipient peer unbundles the package and applies the contents of the package to its file system.

Return Receipts

After a peer opens and executes a package, the results of the package delivery and execution are included in a return receipt that is sent to the publisher that originated the delivery. The publisher can then open and view the return receipt to obtain information about the package delivery and execution. Additionally, the status of the delivery (Pass, Fail, Pending) is reported to the publisher that originated the delivery. All return receipts are logged to the publisher's database.

Delivery Mechanism

Everserve uses publish/subscribe messaging for package delivery, and point-to-point messaging for community management and return receipts from targets. Each community manager, publisher, and relay peer runs a JMS server for delivery of packages among the peers in the community. Each peer must subscribe to at least one JMS server for each community with which it is a member. That is, a target must be associated with either a relay or a publisher, and a relay must be associated with a publisher. When a community is initially created, a universal unique identification (UUID) is generated and bound to that community, from which the JMS server identifier is derived.

The publisher and the package sequence number (each package sent by a publisher has an ever-increasing package sequence number) is set for messages to ensure secure transmissions between publishers, relays, and targets.

Important items to note about this method of package delivery:

- When a package is delivered to the JMS server, only the publisher and the queue provider can identify the targets.
- No target knows of the existence of any other target.
- Each package is encoded for each specific peer and can be opened only by the targets to which the publisher sent the package.

When a publisher initiates a package delivery, the following events take place:

1. The package is created on the publisher.
2. The publisher creates a unique file name of the package specification.
3. Using the package specification file name, the publisher initiates package delivery to the recipient peers.
4. The package is sent to the JMS queuing system for delivery to the appropriate peers.
5. After opening and executing the package contents, each peer sends a return receipt to the publisher, which are then logged in the publisher's database. Return receipts use the same encryption method as packages sent to peers.

Durable Subscribers

As soon as an Everserve peer (target or relay) is running, it becomes a durable subscriber to the JMS queue of the sender (either a publisher or relay) to which they are associated. All peers associated with the senders' JMS queue become authorized subscribing peers, at which point each peer does the following:

1. The subscribing peer retrieves the package from the queue without committing the de-queuing operation. The de-queue operation cannot be committed until the package is opened and the operations within the package are carried out. The queue will retain a persistent copy of the package until all durable receivers commit the de-queuing of the package.
2. The digital signature of the publisher is verified using the peer's public key.
3. The peer finds the symmetric encryption key, using its own private key. The package is then decrypted using the session key.
4. The package is opened and all commands and scripts within are executed.
5. After all recipients have opened their package, the package is removed from the queue.

The results of opening and executing the package are sent back to the publisher using a process similar to the process used for sending a package, with one difference: instead of encrypting the package using the community's symmetric encryption key, the package is encrypted using the publisher's public key and signed by the peer's private key. This response serves as a return receipt for the package, and is sent back to the publisher using a point-to-point queue. This ensures that only the publisher can decrypt the results of the package execution.

Maintaining Persistent Data

Everserve stores all information about communities, peers, and package information in an internal persistence store. This store is represented by a Java interface, and employs two separate implementations; a file store on each target device, and a relational database on each community manager, publisher, and relay.

File Store

A simple file store is used by targets to maintain the data logged during a delivery. The file store keeps all information about package delivery and execution in a single file on the target's file system.

Relational Database

A relational database is used by each community manager, publisher, and relay. The database is accessed through JDBC, and scales to support large numbers of targets. The database is tolerant of changes to the underlying Java classes. Upgrade scripts can be written using SQL operations that directly manipulate the database, and tools may be built on top of the database schema to enable report generation.

Database Configuration

The database is independent of the Everserve virtual machine. The actual database used is specified in a configuration file, allowing database selection and configuration at installation time. By default, the database runs on the same machine as the Everserve device.

Checkpoint Restart

Package delivery can be interrupted due to network failures, hardware malfunctions, or if the connection to the Internet is lost. Connections can be lost at either end of the wire due to a system crash or scheduled maintenance. Checkpoint restart is the process of sending the unsent or non-received portion of a package whose initial delivery was interrupted during transmission.

Once a community manager has added a target to a community, the target must first join the community (on its own server) before it can begin to receive packages. Any target that has not joined the community will not receive packages. Once the target joins the community and becomes a trusted member, it will receive future packages from the publisher or relay. Essentially, unjoined targets are not included in the list of targets to which packages are delivered, nor are packages held in queue for later retrieval.

Packages Received by Targets

Targets process packages in a two-pass approach. The first pass copies the message input stream to temporary disk files and verifies the publisher's digital signature. The target stores the compressed, encrypted stream in the temporary file. Then, if the digital signature is valid, the temp file is read, decrypted, decompressed, and processed.

Checkpoint restart requires temporary space on the target equal to the total size of the compressed package. As portions of the package arrive they are stored and retrieved locally, and only that portion of the package that never arrived at the target is retrieved from the JMS queue when a connection to the target is restored.

If a target crashes in mid-stream of receiving a package, when the target comes back online, it will determine that it was in the middle of receiving a package. It then reads the temporary file to verify the digital signature, then begins reading the rest of the package from the JMS queue.

Delivering Packages to Off-line Targets

Everserve enables the delivery of packages to a target that is off-line at the time of delivery. In this case, any package sent to joined targets that are not available to receive packages are held in queue. All packages held in queue for that target are immediately delivered to the target once the target is back on-line. Once the package is delivered to the target, the target executes the package and sends a return receipt back to the publisher. Packages are executed in the order sent by the publisher if more than one package is held in queue.

Packages Sent from the Publisher

Publishers process packages they send out in a one-pass approach, that is, the system does not write the outgoing ZIP stream to a temporary file on the local file system. If the ZIP stream is interrupted during delivery to the JMS queue, the system will attempt to resend the entire package once a connection to the JMS queue is re-established, rather than re-sending a partial package.

Index