Everserve™

Publisher's Guide

Part No. 2.0-1.12-PUB-01

Table of Contents

Introduction

This *Publisher's Guide* provides information and systematic instructions on how to create package specifications used to effectively deploy applications and content, and deliver packages to massive numbers of globally distributed devices.

This guide addresses the tasks a System Administrator performs to create and publish packages to community peers, and to monitor the delivery status of all package deliveries.

For an introduction to Everserve, refer to the *Everserve Product Overview Guide*. For information on how to install Everserve, how to create and maintain communities, or for tips on system maintenance and troubleshooting, refer to the *Everserve System Administrator's Guide*.

The topics discussed in this Introduction include:

About this Guide

This *Publisher's Guide* is designed to help you understand the principle tasks of creating package specifications, delivering packages, and viewing return receipts.

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, and provides information about additional resources.

Creating and Delivering Packages provides information on how to create package specifications, and deliver packages to Everserve community peers.

Publishing Using the Command Line Interface describes how to use the command line interface to delivery packages to Everserve devices. This section also describes how to view return receipts from the command line interface.

Publishing with Everweb describes how to create and deliver a package specification using Everweb, Everserve's Web-based graphical user interface. This section also describes how to view return receipts from the graphical user interface.

Appendix A: Summary of Everserve Command Line Syntax lists all commands and corresponding parameters that can be issued from the command line interface.

Appendix B: 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 used to create and distribute Everserve packages. 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.

Technical Support

Technical support is available by:

- Calling (831)247-3983
- Accessing Synchron's Web site at http://support.synchronnetworks.com (Login required - see your cover letter for username and password)
- 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://support.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 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	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..
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 or another relay. 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.

Creating and Delivering Packages

Package delivery is the process of sending **packages** containing data and commands from a **publisher** to a set of recipient **target** machines. This section describes how packages are created and delivered to targets in the community.

Although this information on package creation and delivery is generic in concept, the examples and instructions in this section are specific for command line operation. For instructions on how to create and deliver packages using Everserve's Web-interface, see the section Publishing with Everweb.

The topics in this section include:

- What is a Package Specification?
- Creating a Package Specification
- Delivering Packages

What is a Package Specification?

A package specification is an XML file that contains the list of files, applications, datasets, and commands 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.

Package specifications include the following attributes:

- The spec-version of the package specification (indicates forward and backward compatibility with packages created for use with Everserve).
- Command-spec commandline entries that are used to perform OS specific commands on the target (such as "dir" for Windows systems, or "ls" for Solaris systems).
- Directory-spec entries that are used to copy directories (recursively or not) from the publisher's file system to the target's file system.
- File-spec entries that are used to copy files from the publisher's file system to the target's file system.
- Command-spec file entries used to launch executable files on the target (such as scripts, .bat, or .exe for example).

Before packages can be created and distributed, it is assumed that the community has been configured with a publisher. This means that the certificates and configuration information have been distributed to the community peers. Once this setup has been performed, a publisher can deliver packages to the community of peers.

File System Access

Great care and caution should be exercised when creating package specifications used to overwrite or delete files on target systems. When running Everserve as root or Administrator, scripts containing commands to delete or overwrite files on target systems can be included in the package. For example, packages containing scripts and commands to overwrite a Windows targets' autoexec.bat file, or to delete the /var/mail on a Solaris device can be sent to targets, thus the target will apply the package (scripts and commands) to its file system.

For information on how to prevent certain directories or files from being overwritten, see the section System Configuration in the *System Administrator's Guide*.

Creating Packages for Heterogeneous Communities

Everserve packages can be deployed to a heterogeneous community of targets. When a target receives a package, it opens the package and executes the scripts and commands contained in the package. 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 Solaris devices.

Any system that fails to apply and/or execute a package in its entirety will report a "failed" package delivery back to the publisher. Therefore, it is highly recommended that operating system specific packages be created for the corresponding community devices.

For samples of OS specific package specifications, visit Synchron's Web site at http://support.synchronnetworks.com/samples.

Package Delivery Sequencing

Every time a publisher sends a package out, the publisher increments a sequence number associated with that package specification. All peers keep track of the last sequence number they received for a given package specification. This ensures that packages are opened in proper sequence and are opened in the order in which they are sent.

Sample Package Specifications

The following package specifications are simple examples of commands and scripts that can be included for OS specific targets. In each example, the package specification contains commands that will retrieve a directory listing from the target's file system, deliver a directory recursively to the target, deliver an individual file to the target, and then run a file (batch file on Windows, executable file on Solaris) on the target. **Note that all commands are executed in the order presented in the package specification.** The results from opening and executing the package are sent back to the publisher in the form of a return receipt. For example, if you include a command to list a directory listing of a device, this listing will be included in stdout of the return receipt.

Example: Windows Package Specification

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE spec-container SYSTEM "package_spec.dtd">
<spec-container name="SamplePackage" version="false" delta="false" >
     <spec-version version="2.0" earliest="1.0"/>
     <!-- Executed in this order -->
     <!-- Create a directory listing -->
```

```
    <command-spec commandline="dir c:\directory" success-codes=”0” />
      <!-- Copy the directory recursively -->
    <directory-spec source="c:\source\directory" target="c:\target\direc-
    tory" recurse="true" />
      <!-- Copy \source\file to \target\file on peers -->
    <file-spec source="c:\source\file" target="c:\target\file" />
      <!-- Run file.bat on peer -->
    <command-spec file="c:\directory\file.bat" success-codes=”0” />
</spec-container>
```

Example: Solaris Package Specification

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE spec-container SYSTEM "package_spec.dtd">
<spec-container name="SamplePackage" version="false" delta="false" >
     <spec-version version="2.0" earliest="1.0"/>
     <!-- Executed in this order -->
     <!-- Create a directory listing -->
    <command-spec commandline="ls -l" success-codes=”0” />
     <!-- Copy the directory recursively -->
    <directory-spec source="/source/dir" target="/target/dir"
    recurse="true" />
     <!-- Copy /source/file to /target/file on peers -->
    <file-spec source="/source/file" target="/target/file" />
     <!-- Run scriptToRun on peer -->
    <command-spec file="/home/scriptToRun" success-codes=”0” />
</spec-container>
```

For additional samples of package specifications, visit Synchron's Technical Support site at http://www.synchronnetworks.com/support.

Creating a Package Specification

Package specification templates are provided with Everserve and can be used to create custom package specifications. Everserve provides two OS specific package specification templates (*WindowsPackageSpec.xml* and *UNIXPackageSpec.xml)* that can be used to create package specifications. Additionally, Everserve provides an non-OS specific package specification (SimpleTest.xml) that can be sent to any peer regardless of the peer's underlying OS. The default installation location for these templates is on the publisher's file system in the `\Synchron Networks\Everserve\server\Packages` subdirectory.

The information and systematic instructions provided in this section describe how to create a package specification. Because package specifications are XML files, experience with XML is helpful, but not required. For examples of package specifications, see Synchron's Technical Support Web site at http://support.synchronnetworks.com/samples.

Summary

Creating a package specification requires some amount of planning and preparation to ensure appropriate and effective package delivery. The following list summarizes the steps and processes used to successfully create and deliver a package:

1. Scan all files to be included in the package for viruses.
2. Create the scripts you want to include in the package specification.
3. Copy, rename, and modify the *WindowsTemplateSpec.xml* or *UnixTemplateSpec.xml* file and list all scripts, commands, files, and directories you want to bundle together in the package.
4. If delivering file system updates and changes, ensure you have the *VERSION=* and *DELTA=* flags set to *TRUE*.
5. Deliver the package.

Package Spec Elements

Each package specification contains specific XML elements and attributes to define exactly which files, folders, commands, scripts, and batch files to deliver. When the deliver command is issued to deliver a package, Everserve compares each XML element and attribute in the package specification against the *package_spec.dtd* file. This file (*package_spec.dtd*) contains a list of all elements and attributes that can be used in an Everserve package specification.

The following table lists the elements of a package specification:

Element	Example	Description
xml version	<?xml version="1.0" encoding="UTF-8"?>	XML header that identifies the XML version and XML encoding standard.
!DOCTYPE spec-container	<!DOCTYPE spec-container SYSTEM "package_spec.dtd">	XML document type. The spec-container element indicates a collection of system level commands, directories, and files specification elements for the package. These elements are compared to and verified by the *package-spec.dtd* file when a deliver command for the package specification is issued.
spec-container name	<spec-container name="Sample Package" version="false" delta="false" >	Descriptive name for the package (not the <*package_filename.xml*>) which will be shown as the package ID for the delivery when viewing the delivery and return receipt. The spec-container name attribute also includes the *version*= variables used for delivering changed data points only during a delivery. These variables are described in Step 4 below.
spec-version	<spec-version version="2.0" earliest="1.0"/>	Indicates the backwards compatibility of the current package specification. In the example provided, the current package specification will be compatible with systems running Everserve v1.0 (or newer).

Element	Example	Description
command-spec command-line	<command-spec commandline="dir c:\temp" success-codes = "0"/>	Used to issue an OS specific command line directive. In the (Windows) example provided, Everserve will issue the "dir" command against the directory "C:\temp." A success code of zero (0) is the default and may be changed to allow the package to continue executing if/when a non-zero code is possible or expected. See Success Codes for information on how to specify success or failure of an execution of a command-spec element. The standard output and standard error of this operation are included in the receipt that is sent back to the publisher.
directory-spec	<directory-spec source="c:\source\directory" target="c:\target\directory" recurse="true" />	Used to copy a directory form the publisher (source) to a location on the target (target). The variable *recurse* specifies if all subdirectories under the directory specified are to be copied.
file-spec	<file-spec source="c:\source\file" target="c:\target\file" />	Used to copy a file from the publisher (source) to a location on the target (target).
command-spec file	<command-spec file="c:\directory\file.bat" success-codes="0"/>	Used to execute a script or batch file on the target system. This batch or script file is first copied from the publisher to the target, then executed. Once execution completes, the file is removed from the target's file system. A success code of zero (0) is the default and may be changed to allow the package to continue executing if/when a non-zero code is possible or expected. See Success Codes for information on how to specify success or failure of an execution of a command-spec element.

There are several predefined XML characters that may appear in their literal form only when used as markup delimiters, or within a comment. If they are used in a command-spec line, they must be escaped using the delimited substitute character strings listed in the following table. If any of these XML characters are not delimited properly, a parsing error will occur when the target executes the package specification.

XML Character	XML Delimited Substitute
&	&amp
<	&lt
>	&gt
“	&quot
‘	&apos

Package Specification Error Handling

The way in which Everserve handles package specification errors depends on the type of error in the package specification. If a package specification contains an XML usage error, no attempt to deliver the package is made and a “sax parse” error is generated. Similarly, if a file in a file-spec command does not exist or cannot be found on the publisher's file system, an error is displayed and no delivery is attempted. If a package specification contains command-spec elements that will not be recognized by the target's OS (such as “Dir” on a Solaris system, or “ls” on a Windows system), Everserve attempts to deliver the package, however, the package delivery will fail and an error will be noted in the delivery receipt.

Using Environment Variables in Package Specifications

Everserve lets you incorporate environment variables within package specifications so that system variables are used on the target peer. When environment variables are used in the package specification, and the package is delivered to a target, Everserve first looks at the environment variables on the target peer to locate the variable called out in the package specification. If the environment variable is found, Everserve expands the variable (on the target) called out in the package specification and completes the command or file execution. If it can not find the variable specified, it then searches the Java System Properties for a match. If no match can be found from either attempt, the variable is not expanded, resulting in an error.

For example, if you include a command in the package specification to copy a file to the target's root directory, you can specify the system variable "SYSTEMROOT" in the macro. When the package is opened and executed by the target, Everserve copies the file to the target's root directory, regardless of where "root" is located on the target (where "root" could be located on drives C:, D:, E:, or wherever root is defined on the target system).

> *Note: Everserve inherits environment variables for the account under which it runs. The default account is "System." Running Everserve under a user account will enable that user account's environment variables.*

If you want to use variables in Everserve packages that refer to Windows environment variables, ensure these variables are declared for the device.

> *Note: Windows does not make all system variables available to Services.*

To Verify Environment Variables on Windows 2000, WinNT, and XP:

1. Right-click **My Computer**.
2. Select **Properties**.
   The system displays the System Properties dialog box.
3. Click the **Advanced** tab.
4. Click **Environment Variables**.
   The system displays two lists: one containing variables defined for the logged-in user, one showing system variables only. Scroll through the System Variables list to determine if the environment variable you want to use in the package specification is supported.

The environment variable is designated as ${<variable>} in the package specification. The following sample package specification illustrates using this technique to copy the file "file.txt" to the root directory of the target peer receiving the package.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE spec-container SYSTEM "package_spec.dtd">

<spec-container name="Variable Example" version="false" delta="false" >

<spec-version version="2.0" earliest="1.0"/>

<!-- Copy file to system root on target-->

<file-spec source="c:\source\file.txt" target="${SYSTEMROOT}\file.txt"/>

</spec-container>
```

Environment variables can be used when specifying commands, file specifications, or any command-spec string that is enclosed in quotations marks (" ").

Note: When executing commands in a package specification (or in a script that is included in the package specification), all commands will be executed on the target in the order listed in the package specification when the package arrives at the target. Each package specification created may contain up to 250 specification attributes.

To Create a Package Specification:

1. Using a text editor or the XML editor included in Everweb, open the package specification template ***WindowsTemplateSpec.xml*** or ***UNIX TemplateSpec.xml*** located on the publisher's system in the `\Synchron Networks\Everserve\server\Packages` directory.
2. Save the package specification with a new file name, for example, *MyPackage.xml*
3. Include the name of the package to be delivered in the ***spec-container name*** string. For example, to specify the package file name "*VirusCheckUpdate:*"

   ```
   <spec-container name= "VirusCheckUpdate" version="false" delta="false">
   ```

4. Use the ***spec-container*** string and set the following parameters as needed:

Version= The VERSION flag is used to specify whether or not the metadata of the files and directories that are bundled together (in the current delivery) is recorded to the database. This feature is useful if you want Everserve to track changes to the files and directories listed in the package specification. When changes are tracked and recorded to the database (VERSION=TRUE), you can then use the DELTA =TRUE flag to deliver only those changes to community peers that previously received the package.

- **True** Metadata of the files and directories listed in the package specification for the current delivery will be recorded to the database.
- **False** Metadata of the files and directories listed in the package specification for the current delivery ***will not*** be recorded in the database.

Delta= The DELTA flag is used to check the state of the file system (files and directories) from the last time the metadata for the package specification was recorded to the database (that is, the last time the package specification was run with the VERSION = TRUE).

- **True** Everserve checks the metadata in the database for file system changes. Only the changes from the last recorded delivery are compared to the current delivery. Then, only those files or directories that have changed since the last delivery are bundled and delivered to the intended peer(s) in the community.
- **False** Sends all files and directories listed in the package specification to the intended peer(s) in the community regardless of the file system's changed state. When DELTA = FALSE is set, Everserve **does not** check the database for file system changes prior to delivery of the package.

Note: The spec-container VERSION and DELTA settings apply to the directory-spec and file-spec elements of a package only.

5. If including OS commands that the target is to execute, use the ***command-spec commandline*** string to include the executable commands you want each receiving peer to execute. To have the system return a numerical code value upon the successful completion of the command, include this value in the command string as shown in the examples that follow. See the section Success Codes for information on how to indicate a "pass" or "fail" status in a receipt for a delivery.

   UNIX:

   ```
   <command-spec commandline="ls -l" success-codes="0" />
   ```

   Windows:

   ```
   <command-spec commandline="dir" success-codes="0" />
   ```

   Note: UNC notation can be used on Windows systems when the Everserve service is run under an account with rights to shared network locations.

   ***Double quotes must be used** when specifying a command-spec for a commandline tag that includes spaces in the directory name. Failure to enclose directory paths that use spaces for a commandline tag will result in a failed delivery. For example:*

Example, Correct Syntax:

```
<command-spec commandline="dir "C:\Program Files\data\HR forms.file"" />
```

Example, Incorrect Syntax:

```
<command-spec commandline="dir C:\Program Files\data\HR forms.file" />
```

It is important to note that including the Everserve ***STOP*** command in a package specification will stop the target, but a return receipt ***will not*** be sent back to the originating publisher indicating that the device has been successfully stopped, or that the device ever received the package at all. Since Everserve is essentially stopped and not running once the device receives a STOP command, the device cannot complete its processing cycles, that is, it cannot send the return receipt back to the publisher because it is no longer running an Everserve process.

Success Codes

Success codes are used to indicate the "pass" or "fail" status of a delivery. This value is user defined and can be one or more numbers, a numerical range, or "All." The following table describes the various Success Code settings supported:

Success Code Value	Description
0	If target experiences an error level of 0, then all actions performed on the target passed. Otherwise, a return value of anything other than 0 indicates a failure on the target device.
0-10	If the return success code is any value less than 11, then all actions performed on the target passed. Any numerical value greater than 10 (11 or higher) indicates that an action failed on the target.
All	Using "All" as a success code will return a "pass" status for the delivery regardless of errors (if any) experienced on the target when executing the commands listed in the package.

6. If replicating an entire directory to a target, use the ***directory-spec*** string and change the parameters as follows:

Source Full path of [source] directory to which to propagate to the targets.

Target Full path of [target] directory to which to add or copy content on the targets.

Recurse True to replicate the entire directory structure of source directory. False to replicate only the directory specified.

Note: The relative directory path for each directory-spec element begins from the \Everserve\server directory.

For example, to the copy the entire "HRForms" directory and all subdirectories under "HRForms," from the publisher's device to the target device:

```
<directory-spec source="C:\HR2002\HRForms" target="C:\HR2002\HRForms"
   recurse="true" />
```

To copy the entire "HRForms" directory only (without copying all subdirectories under "HRForms") from the publisher's device to the target device:

```
<directory-spec source="C:\HR2002\HRForms" target="C:\HR2002\HRForms"
   recurse="false" />
```

Note: Wildcards are not permitted in package specifications.

***Do not use double quotes** when specifying a directory-spec that includes spaces in the path or file name. Doing so will result in a failed delivery. For example:*

Example, Correct Syntax:

```
<directory-spec source="C:\Program Files\Data\HR forms.txt" tar-
   get="C:\usr\local forms\HR Forms.txt" recurse="false" />
```

Example, Incorrect Syntax:

```
<directory-spec source=""C:\Program Files\Data\HR forms.txt"" tar-
   get=""C:\usr\local forms\HR Forms.txt"" recurse="false" />
```

7. If replicating individual files to the target, use the ***file-spec*** string and change the parameters as follows:

*Note: If more than one file is to be propagated for this package, you must enter each file you want to propagate as a separate **file-spec** string.*

Source Full path of [source] file name to which to propagate to the targets.

Target Full path of [target] file name to which to copy content on the targets.

Note: The relative file path for each directory-spec element begins from the \Everserve\server directory.

For example, to copy the file "NewEmployeeForm" from the publisher's device to the target device:

```
<file-spec source="C:\HR2002\HRForms\NewEmployeeForm" target="C:\HR2002\
   HRForms\NewEmployeeForm" />
```

Note: Wildcards are not permitted in package specifications.

***Do not use double quotes** when specifying a file-spec that includes spaces in the path or file name. Doing so will result in a failed delivery. For example:*

Example, Correct Syntax:

```
<file-spec source="C:\Program Files\Data\HR forms.txt" tar-
   get="C:\usr\local forms\HR Forms.txt" recurse="false" />
```

Example, Incorrect Syntax:

```
<file-spec source=""C:\Program Files\Data\HR forms.txt"" tar-
   get=""C:\usr\local forms\HR Forms.txt"" recurse="false" />
```

8. If including scripts to run on the target, use the ***command-spec file*** attribute to include any batch files or scripts you want each receiving peer to run. For example:

   ```
   <command-spec file="/home/FileToRun.bat" success-codes="0" />
   ```

   If more than one script is to be included in this package, you must enter each script you want the peers to run as a separate **command-spec file** string.

   Scripts are copied to the `\Synchron Networks\Everserve\server` directory on the target device, executed, then automatically deleted.

9. Save the package specification.

   Note: The default location for packages specifications is `\Synchron Networks\Everserve\server\Packages`. *You may save package specifications to a directory other than the default. When doing so, you must also copy the package_spec.dtd file from the default directory to the same directory as the package specification before attempting to deliver a package.*

Delivering Packages

Everserve's delivery mechanism allows package delivery to all peers in a community or to a list of peers in a community. Before delivering a package, make sure all files included in the package are free from computer viruses. Failure to do so may result in the propagation of an undetected virus to all peers in an Everserve community.

Testing Packages Prior to Delivery

Before delivering a package to any peers, **it is strongly recommended that all packages be tested** for potential delivery problems. This will help to identify any errors in the package that can be resolved before package delivery, resulting in clean package executions on the community targets.

To test a package, create a test community and deliver the package to this test community. This can be as simple as creating a target peer on the same device as the publisher, then delivering the package to this test target. Although this type of test community can not identify potential routing problems that can occur when devices are networked, it can detect most execution problems that occur on the target due to errors in the package specification such as command-string sequencing, missing files or directories, or batch and script file errors.

Once you have tested a package, you are ready to deliver the package to the community peers. The information in the following sections describes how to deliver packages to all peers in a community and to how to deliver a package to individual (one or more) peers in a community.

Delivering a Package

The following section describes the delivery syntax used to deliver packages to an entire community, and to a list of peers in a community.

Note: By default, package specifications (<packageFileName.XML>) reside in the `\Synchron Networks\Everserve\server\Packages` directory on the publisher's file system. This is also the default location of the XML DTD file as well. If package specifications will reside in a directory other than the default, you must copy the DTD file to the same directory where the package specifications are located.

Command Syntax:

```
everserve deliver [immediate]{-f <packageFileName>} [-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 <packageFileName>}	Y	Name of the package you wish to publish to the community. If path is not specified, Everserve will look in the \Packages directory for the package specification.
[-c <communityName>]	N	Name for the community. If more than one community exists on the device you must specify the community name. Community names are case sensitive.
[-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.

Syntax	Required	Description
[-i <ID>]	N	The ID of the package is used to create a familiar name for the package, such as "Update_2002". This ID is also used as identifier that accompanies the packages so that the user can distinguish package deliveries.
[-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, as defined by the list of peer names specified, separated by spaces. If a relay is specified as the delivery recipient, all peers connected to that relay will receive the delivery. Peer names are case sensitive.

Delivering a Package to an Entire Community

The following example illustrates how to deliver the package named "*myPackage.XML"* to the community called *"myCommunity,"* using the publisher *"MyCommunityPub."* The delivery command shown also labels the package (using the -i parameter) with the familiar name *"Update_2002"* used to identify the package for this delivery.

```
everserve deliver -f myPackage.xml -c myCommunity -p MyCommunityPub
   -i Update_2002
```

Note: The [-c <communityName>] and [-p <publisherPeerName>] parameters for the deliver command are optional if there is only one community to which to publish to, and that community has only one publisher.

Delivering a Package to a List of Peers

There will be instances when one or more peers in the community require delivery of a package, while other peers in the community will not. This is often true when a new peer joins the community and requires a full package update to bring the new peer's file system and applications to the same levels and versions as the other peers in the community. Delivering system upgrades and patches may also require delivering a package to a few selected peers in the community, while other peers in the community may not require these types of packages.

The following example illustrates how to deliver the package named "*WelcomePackage.XML"* to peers "*newPeer1," "newPeer2*," and "*newPeer3*" in the community named "*myCommunity*." The delivery command shown also labels the package (using the -i parameter) with the familiar name *"Welcome_newPeer_Package"* used to identify the package delivery.

```
everserve deliver -f WelcomePackage.xml -p MyCommunityPub -c myCommunity
   -l newPeer1 newPeer2 newPeer3 -i Welcome_newPeer_Package
```

Note: The [-c <communityName>] and [-p <publisherPeerName>] parameters for the deliver command are optional if there is only one community to which to publish to, and that community has only one publisher.

Publishing Using the Command Line Interface

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 to the database. Package specifications are XML files created prior to delivery and contains a list of actions that the target devices are expected to perform. Actions can be any task such as receiving files and datasets, install an application, update, or patch, or execute a command. The results of all actions performed by the target are returned to the publisher in the form of a return receipt.

The information in this section describes how to execute publisher commands from the command line interface only. For information on how to create a package specification, see the section Creating and Delivering Packages in this guide.

The topics discussed in this section include:

- Using the Interactive Command Shell
- Delivering Packages
- Viewing Return Receipts
- Executing Commands in Batch Mode

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 2.0-1.12 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."

Delivering Packages

Packages may be stored in a default directory (`\Synchron Networks\Everserve\server\Packages`)on the publisher's machine. Alternately, packages may be stored in a location other than the default provided. See the section Creating and Delivering Packages for information on managing package specifications.

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

Command Syntax:

```
everserve deliver [immediate]{-f <packageFileName>} [-c <communityName>]
   [-p <publisherName>] [-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 <packageFileName>}	Y	Name of the package you wish to publish to the community. Package names are case sensitive.
[-c <communityName>]	N	Name for the community. Community name strings are alphanumeric and can include symbols.
[-p <publisherName>]	N	Name of the publisher that has control or responsibility for publishing the specified package. If more than one publisher exists in the community, you must specify which publisher is to deliver the package.
[-i <ID>]	N	The ID of the package is used to create a short, familiar name for the package, such as "update_2002."
[-l <peerList>]	N	By default, the delivery command normally delivers to all targets connected to the publisher. The -l switch enables delivery of a package to one or several peers, as defined by the list of peer names specified. Multiple peer names are separated with a space. For example: -l peer1 peer2 peer3

Examples:

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

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

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

```
everserve deliver immediate -f FormsUpdate -p HRpub -l peer1 peer2 peer3
   -i "Update_2002"
```

Viewing Return Receipts

A return receipt is a manifest of the results of opening a package that was delivered to a target. The contents of the return receipt include the results of applying file operations, standard output, standard error, the return code of all scripts that were executed as a result of opening a package, and the package specification. All targets that receive a package send a return receipt back to the publisher that originated the package delivery. Return receipts include a time stamp indicating when the package was executed, the status of the package delivery, and other delivery details.

When a publisher sends a package to a target, the target opens the package and executes the commands and scripts contained in the package in the order listed in the package specification. Once the target either executes the package in its entirety, or if it fails to execute any part of the package, a return receipt is immediately sent to the originating publisher indicating completion or failure of the package's execution. All return receipts received from targets are automatically logged to the publisher's database.

> *Note: It is important to note that if a target receives an Everserve **STOP** command in a package specification, a return receipt **will not** be sent back to the originating publisher indicating that the device has been successfully stopped, or that the device ever received the package at all. Since Everserve is essentially stopped and not running once the device receives a STOP command, the device cannot complete its processing cycles, that is, it cannot send the return receipt back to the publisher because it is no longer running an Everserve process.*

Return receipts are viewed on the publisher's system only. All return receipts are located in the database associated with the publisher's system. Return receipts are identified by a unique name (based on the spec-container name contained in the package specification) and a time stamp of when the package was executed by the target.

Return Receipt Contents

Return receipts for each delivery are sent from each target to the originating publisher, and stored in the publisher's database. Each return receipt is identified by the package ID and time stamp when the package was executed. Each return receipt contains the following elements:

- Peer name
- Descriptive package name
- Package specification, package specification file name and path
- Status of the package specification delivery: either pass or fail
- Date/timestamp when delivery was received by the target
- All commands delivered in the package specification
- Status of the command-spec execution: either pass or fail

- Script return code
- Standard output of the package
- Standard error of the package execution

The following sections describe how to view return receipts from the command line interface. For information on how to view return receipts from the Web interface, see Publishing with Everweb.

Accessing Return Receipts from the Command Line Interface

Everserve provides various levels of return receipt information used to monitor delivery status and results of package execution. Using the appropriate command line syntax, you can quickly determine which peers in a community returned a receipt back to the publisher, the status of the package execution for each peer in the community, if the package was received and executed with or without errors, and view detailed information for each peer on how the peer executed the package contents. For examples of viewing receipts, see the section Show Receipts in this guide.

*Note: The **show receipts** command is valid on publisher devices only.*

Command Syntax:

```
everserve show receipts [-a] [-v] [-c <communityName>] [-f <filter>]
   -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.
[-v]	N	Verbose setting to display detailed information about the package delivery.
[-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 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 use the wildcard character (*) for an imprecise match.

Parameter	Required	Description
[-i <ID>]	N	The ID of the package used as a short, familiar name for the package, such as "update_2002." Package ID's allow the use of the wildcard character (*) for imprecise ID matches.
[-v]	N	Verbose setting to display detailed information about the package delivery.
[-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.

Executing Commands in Batch Mode

The ***run*** command is used to execute one or more commands in batch mode. 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 command you wish to execute. Filenames are case sensitive.

Example:

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

1. Create a text file, for example, "HRCom.txt."
2. List the Everserve commands to include, for example:

```
# what peers are in the -c Example community?:
show peers -c Example
# deliver a package:
deliver -f SecurityPatch.xml -c Example -p HRpub -i SecurityPatch_1
# verify delivery was successful:
show deliveries -i SEcurityPatch_1
```

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

```
everserve run -f HRCom.txt
```

   The system runs all commands listed in the file, in the order in which they are listed.

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: • 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: • 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 deliveries 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.xm07/25/02 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 (*).
[-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 using a short, familiar name, such as "update_2002." Package ID's permit the use of the wildcard character (*).
[-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 log files 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."

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
```

Publishing with Everweb

Everweb is a Web-based graphical user interface (GUI) you can use to create and deliver packages. Everweb is accessible from any Everserve system installed with community manager or publisher capabilities. Only those systems that have publisher capabilities installed have access to Everweb's package creation, delivery, and return receipt viewing features.

The topics in this section include:

Connecting to Everweb

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."

To connect to the Everweb user interface:

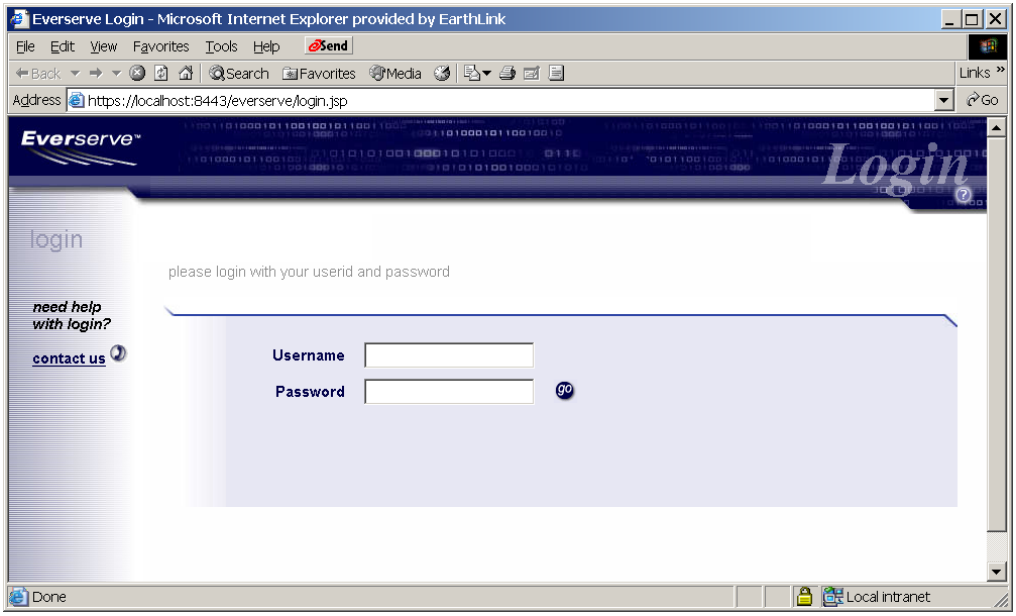
Windows:

From the Windows Start Menu, choose **Programs**> Everserve> **Everweb Session**.

Solaris:

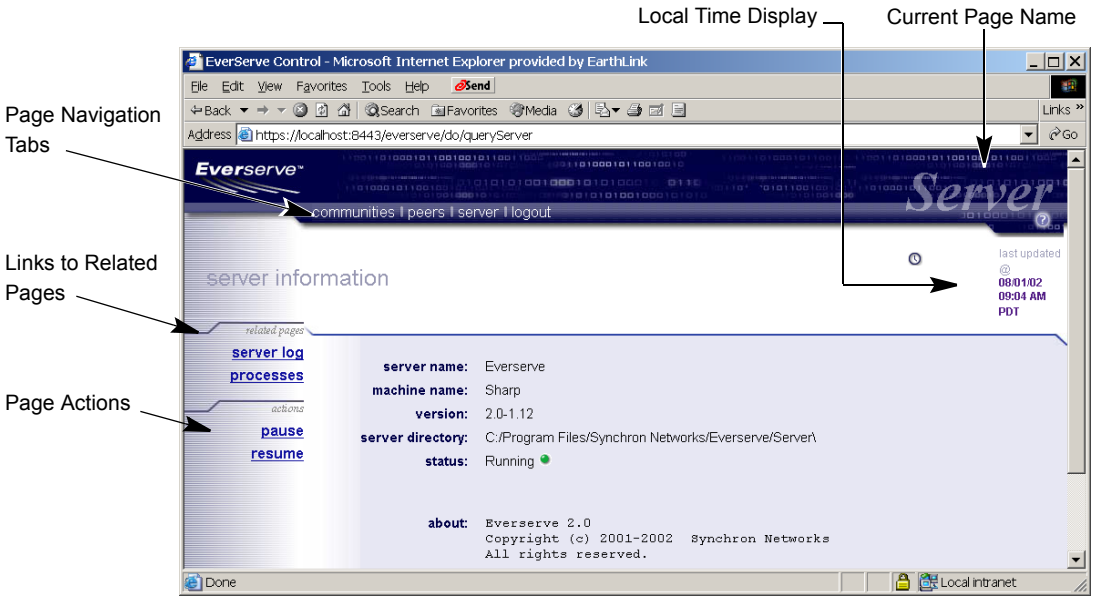
From the command line, type `everweb`.

Once connected to Everweb, the system displays Everserve's Login Web page:



1. Enter your network login name and password and click **Go**.

The system displays the Server page:



Web Page Overview

The top pane of the Web page displays the navigational tabs used to traverse through Everweb pages, 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 are not available.*

The left pane contains links to task related pages and a list of actions that can be performed relative to the current page. The center of the page is used to display, select, 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, and to create community peers.
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 when adding the peer to a community.
Packages	The Packages page is used to view, create, and modify package specifications, and to initiate a package delivery.
Deliveries	The Deliveries page is used to monitor the progress of deliveries and view return receipts.
Server	The Server page is used to access system and process log files, and to pause and resume Everserve on the local device.
Logout	The Logout page is used to log out of Everweb 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:

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 delivery is also displayed.
Red	The Everserve system is not running.	Indicates an error in the delivery for one or more return receipts.

Creating a Package

The information in this section describes how to use Everweb for package creation and modification. It does not provide information or details on the contents of a package specification. For information on package contents, syntax, and format, see the section Creating and Delivering Packages in this guide.

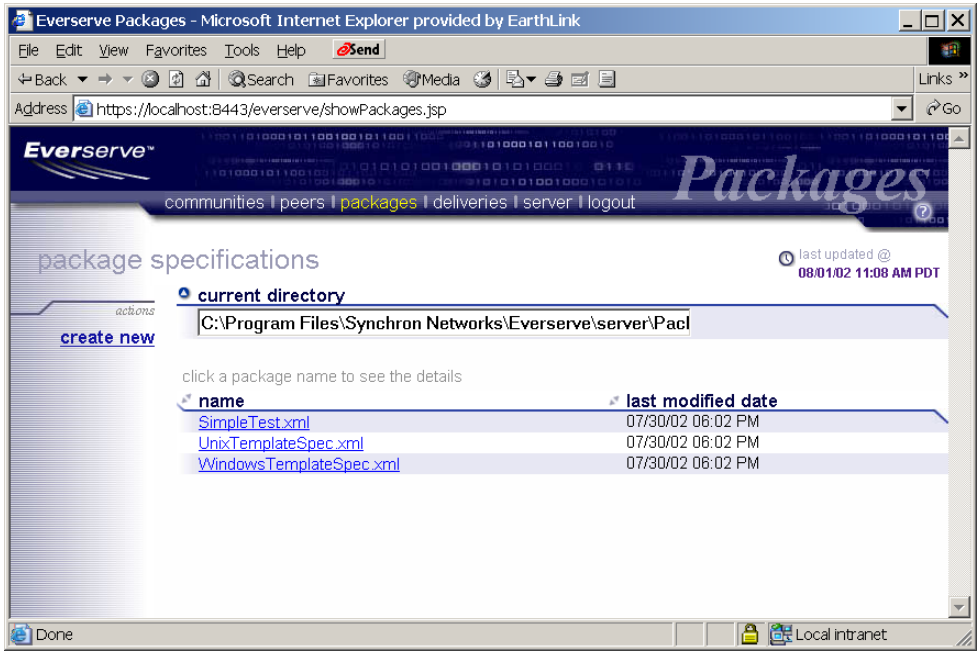
The steps to create a new package specification are identical to the steps taken to modify an existing package specification. The instructions that follow are directed towards creating a new package specification.

To Create or Modify a Package:

1. Click the **Packages** tab.
   The system displays a screen showing a list of all packages that are located in the default package directory *(*Everserve\server\Packages*)*.

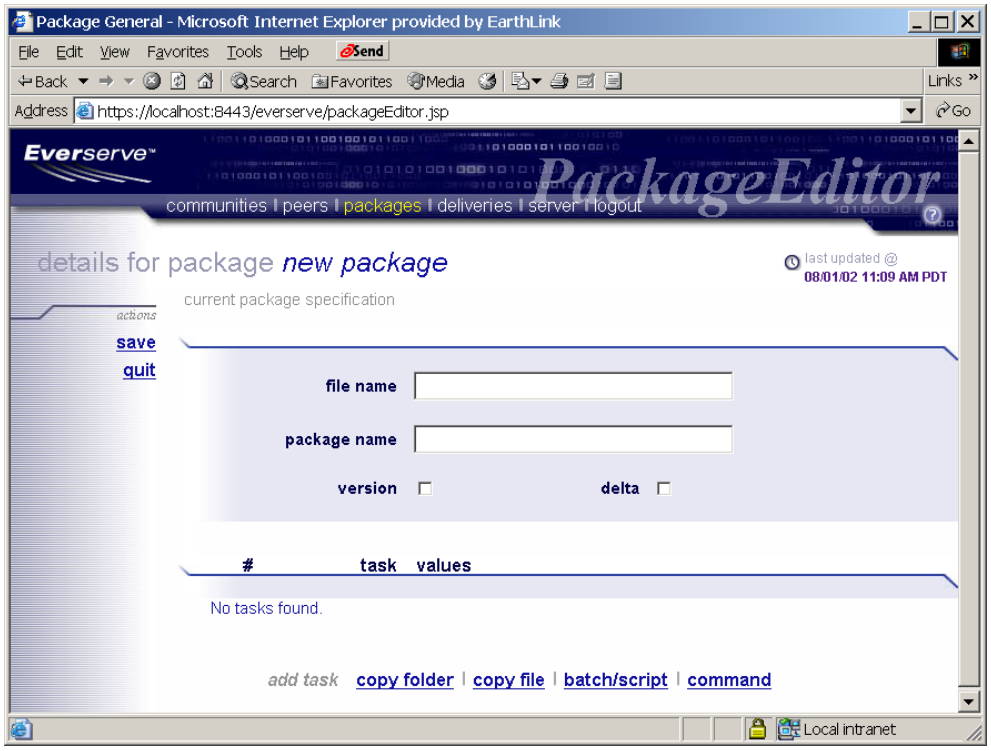
   Note: Package specifications may be stored in directories other than the default directory provided. To view and access packages located in a directory other than the default directory, modify the Current Directory path to point to the directory of choice.

   The following example shows the package specification templates that are located in the default directory:



2. Click **Create New** to create a new package specification.
   Click the **Package Name** you wish to edit an existing package specification.

   The system displays the Package Editor page, as shown in the following example:



3. Enter a file name for the package in the **File Name** field. If you do not include the .XML file name extension, the system will add the extension automatically when the package specification is saved.

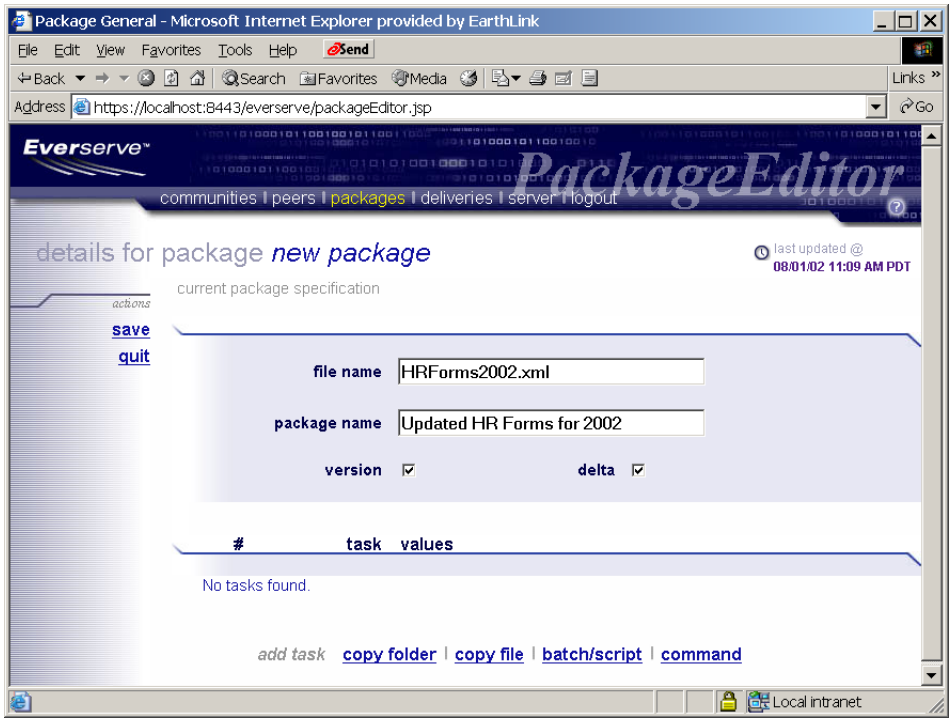
4. Enter a logical name for the package in the **Package Name** field.

5. To record the metadata of this specification to the database, check the **Version** checkbox.

   Note: Version is used to specify whether to record the metadata of the files and directories that are bundled together (in the current delivery only) to the database. This feature is useful if you want Everserve to track changes to the files and directories listed in the package specification. When changes are tracked and recorded to the database, you can then use the DELTA flag to deliver only those changes to community peers for future deliveries of this package.

6. To send file or directory changes only, check the **Delta** checkbox.

   Note: Delta is used to check the state of the file system (files and directories) from the last time the metadata for the package specification was recorded to the database (that is, the last time the package specification was run with VERSION checked). The first time the package is delivered, all files will be copied in their entirety.



7. Click a task listed at the bottom of the Package Editor page to add that task to the package specification. For example, if you want to:
   - Copy an entire directory or folder to peers, click **Copy Folder**.
   - Deliver a file to peers, click **Copy File**.
   - Send a batch file or script to peers, click **Batch/Script**.
   - Have peer systems execute a command, click **Command**.

Package Element Task Ordering

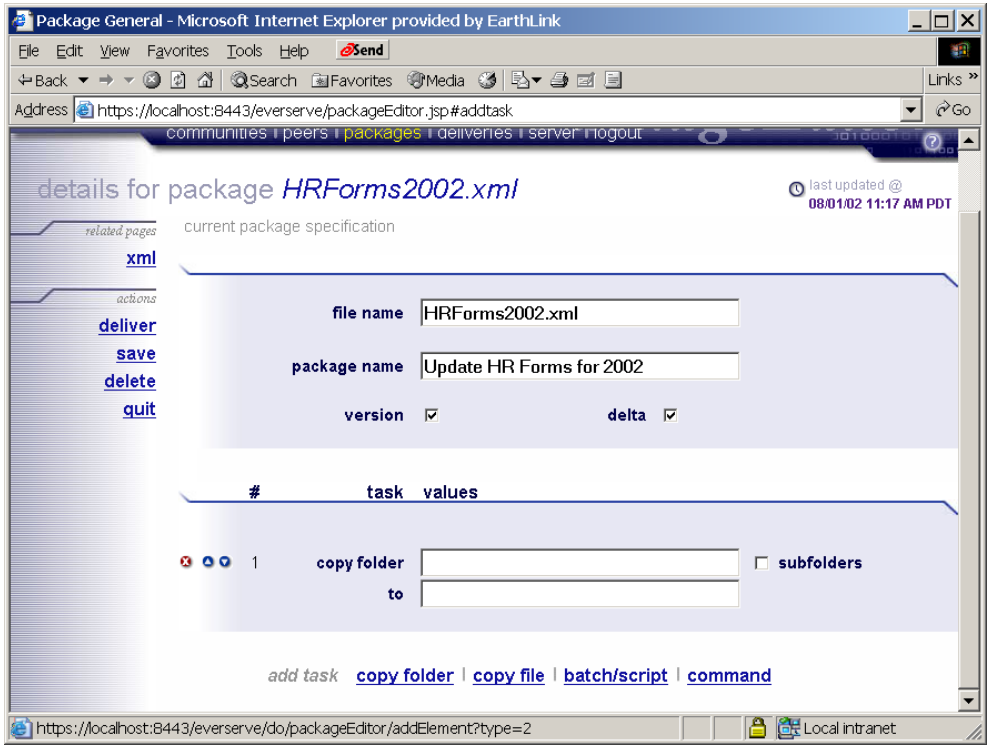
When a package is received by a peer system, it will attempt to execute each entry in the package specification in the order presented. You can rearrange the order of entries in the package by clicking the up/down arrows located next to the task. To delete an entry from the package specification, click the red 'x' located next to the task you want to remove from the package.

Specifying Datasets and Commands

The following sections describe how to specify the datasets to publish to peers, and how to specify which commands to execute on a peer's system. All datasets specified in a package specification must reside on the publisher's local system; all executable commands that are specified in a package specification should only include those commands that the target peer's OS understands and can perform. For additional information about the contents of a package specification, see the section Creating and Delivering Packages.

Adding a Folder to the Package

1. From the Package Editor page, click **Copy Folder**.
   The system displays text fields in which to specify the source folder, the destination in which to copy the folder, and whether to copy subfolders to the target's local file system.



2. Enter the name of the source folder you want to add to the package in the **Copy Folder** text field.
3. If you want to copy all subfolders located directly beneath the folder specified in step 2, check the **Subfolders** checkbox.
4. Enter the destination to which to copy the folder in the **To** text field.
5. Repeat steps 1 through 4 for all folders you want to copy to peers.

Figure 1 shows an example of copying the folder "C:\Docs" and all subfolders within "C:\Docs" to each peer's "C:\Temp\Docs" folder.

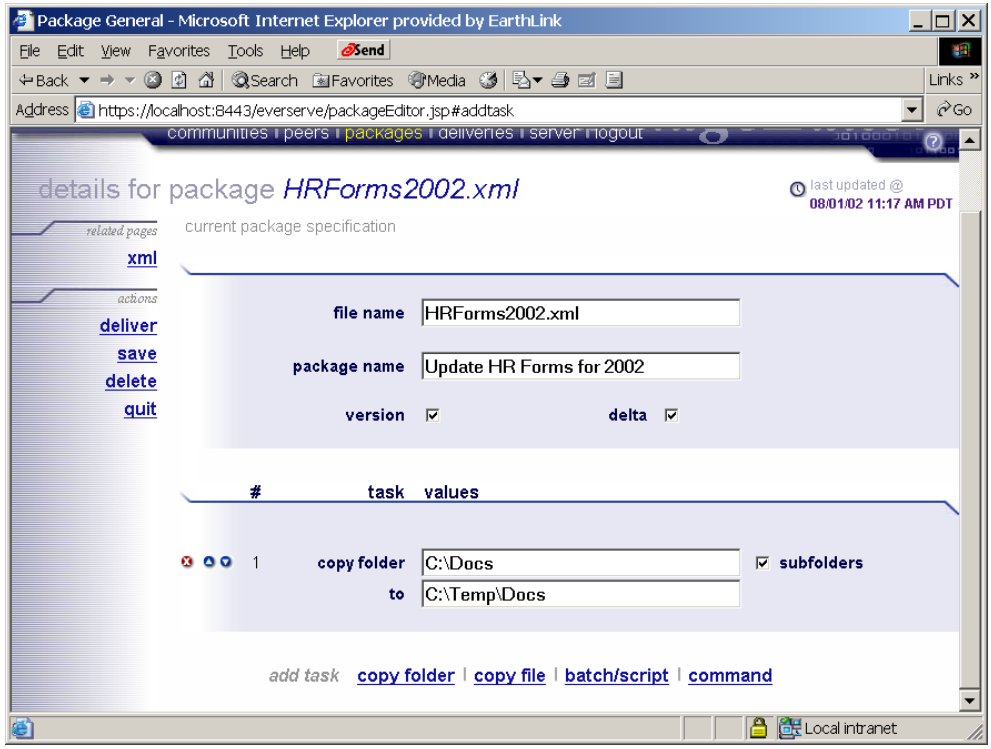
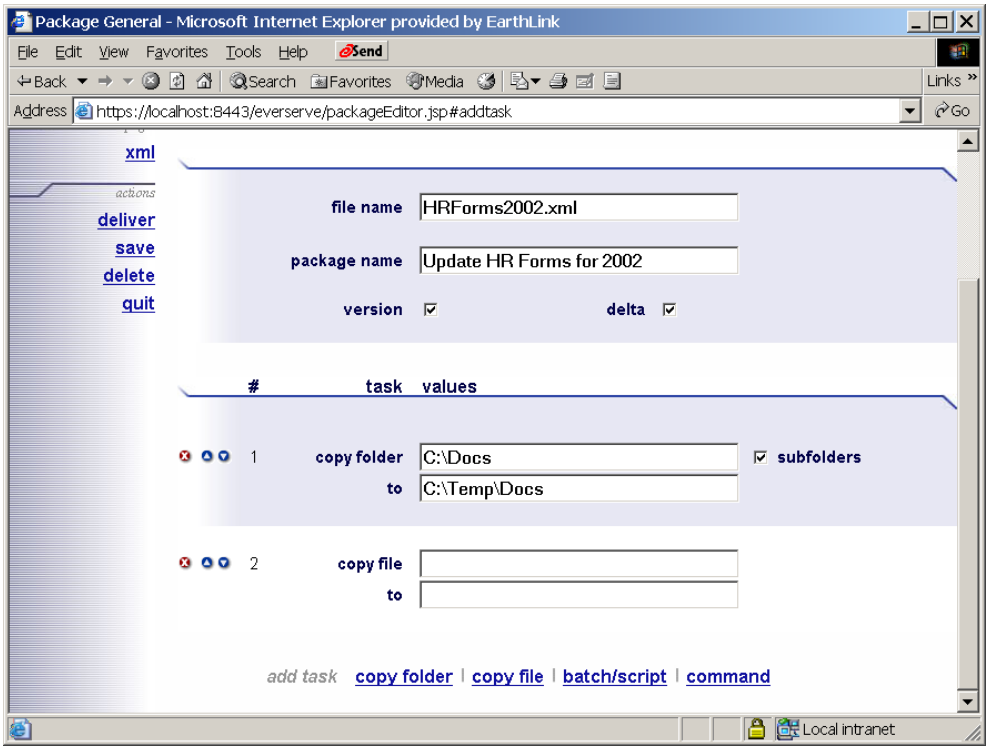


Figure 1 Example, Including Copy Folder Command in a Package Specification

Adding Files to the Package

1. From the Package Editor page, click **Copy File**.
   The system displays text fields in which to specify the source file and the destination in which to copy the file.



2. Enter the name of the source file you want to add to the package in the **Copy File** text field.
3. Enter the destination to which to copy the file in the **To** text field.
4. Repeat steps 1 through 3 for all files you want to copy to the peer's file system.

Figure 2 shows an example of copying the file "HRforms2002.pdf," located in "C:\Forms" folder to each peer's "C:\Forms" folder.

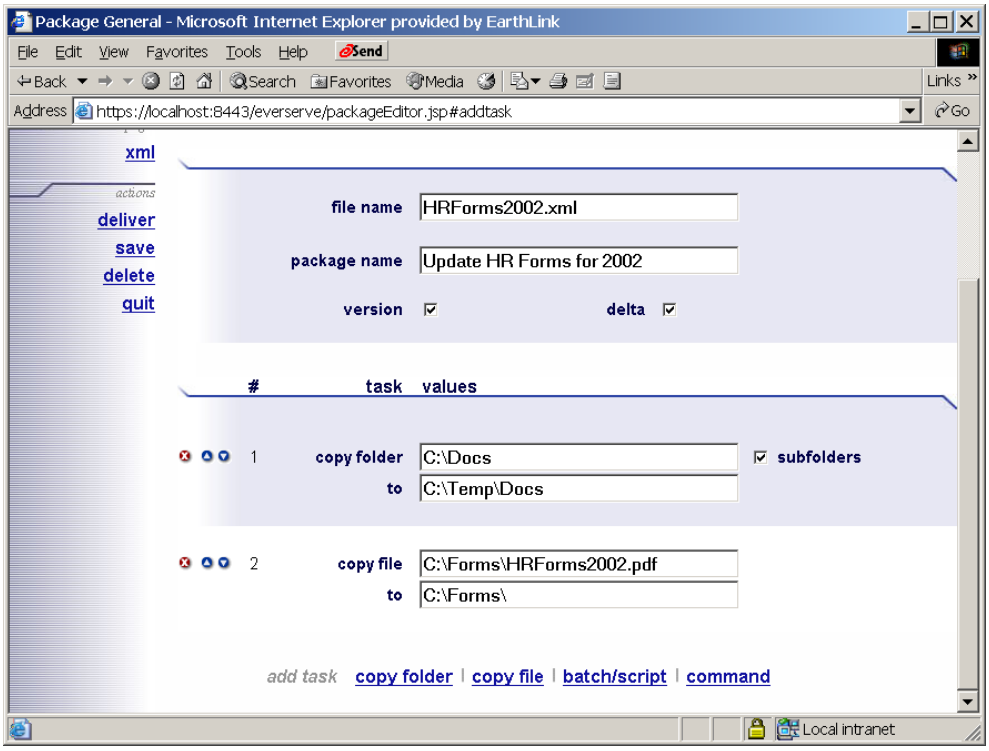
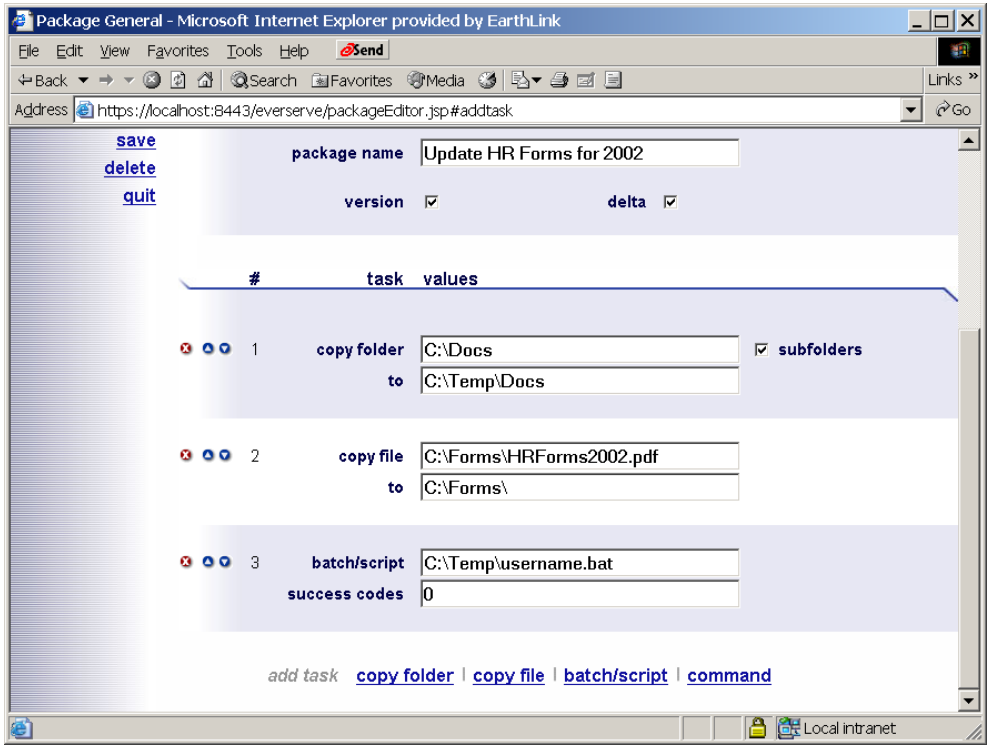


Figure 2 Example, Including Copy File Command in a Package Specification

Adding Batch and Script Files to the Package

1. From the Package Editor page, click **Batch/Script**.

   Everweb displays a text field in which to specify the batch file or script to include in the package, and provides an optional success codes field used to specify a numerical value that indicates a "pass" or "fail" operation:



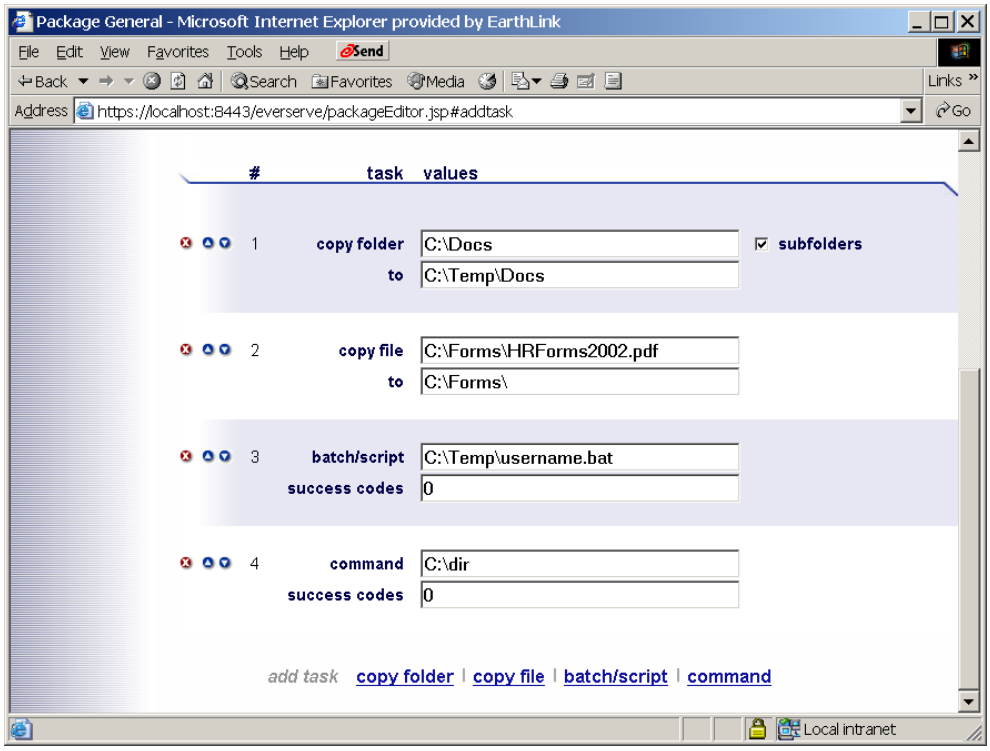
2. Enter the name of the batch or script file you want to include in the **Batch/Script** text field.
3. Enter a numerical value in the **Success Codes** field to indicate successful execution of the script or batch file.

   Note: Success codes are user definable and are used to indicate a pass or fail state of the delivery. See the section Success Codes in this user guide for examples of using success codes in package specifications.

4. Repeat steps 1 through 3 for all batch files and scripts you want to run on the peers.

Adding Executable Commands to the Package

1. From the Package Editor page, click **Command**.
   The system opens a text field in which to specify the command to run on the peers:



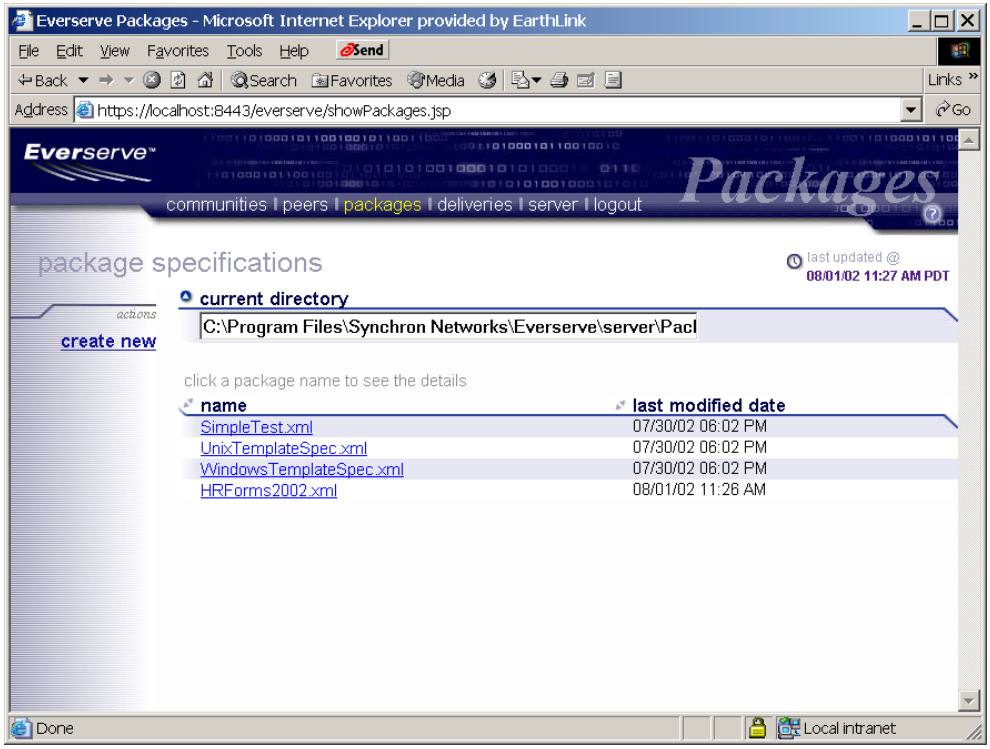
2. Enter the command you want to include in the **Command** text field.
3. Enter a numerical value in the **Success Codes** field to indicate successful execution of the command.
4. Repeat steps 1 through 3 for all files you want to run on the peer devices.
5. Click the **Save** link in the left pane to save the package specification.

Delivering a Package

Deliveries can be made from the Packages page only. The instructions in this section step through the process of initiating a package delivery from a list of available packages displayed on the Packages page.

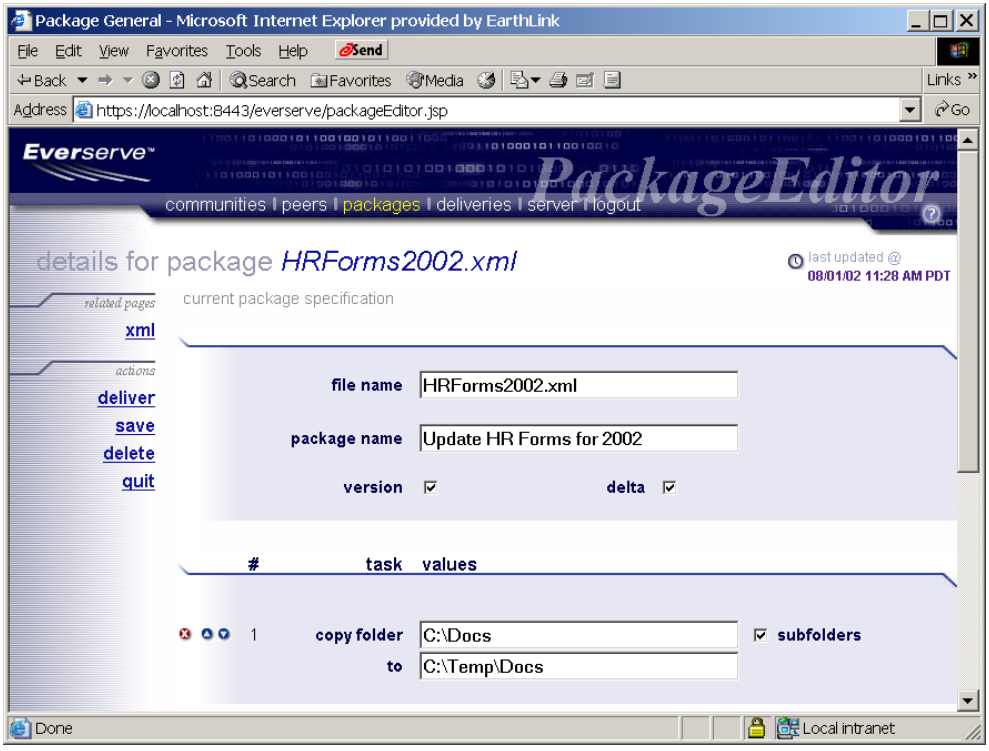
To Deliver a Package:

1. Click the **Packages** tab.
   The system displays the Packages page:

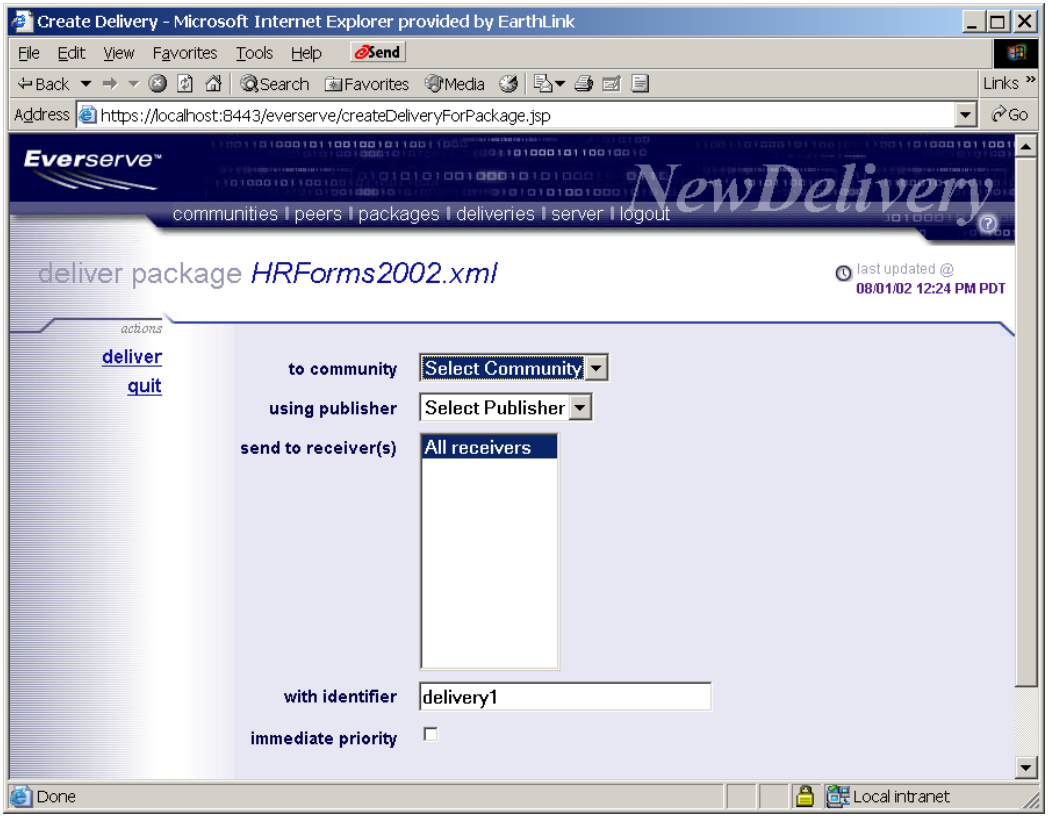


   The Packages page lists all package specifications available from the current directory. To view packages in a different directory, specify the directory path in the **Current Directory** field.

2. Click a **Package Name** you wish to deliver.
   The system displays the Package Editor page.



3. Click **Deliver**.
   The system displays the New Delivery page from which to choose the community that will receive the package, select the publisher to use to send the package, and select which community peers will receive the package. You can also use the default package ID provided, or enter a unique ID used to identify the package's delivery status and return receipt.

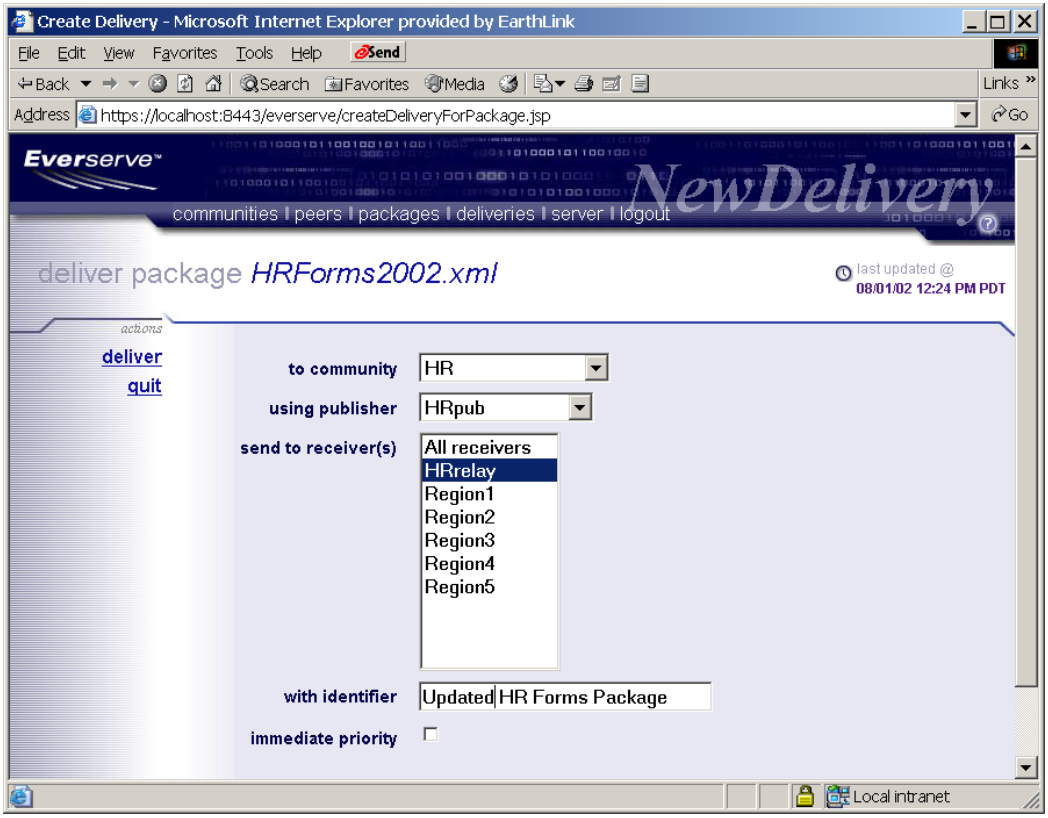


4. Select the community to which to deliver the package from the **To Community** pull-down list.
5. Select the publisher that will send the package from the **Using Publisher** pull-down list.
6. Select the peers in which to deliver the package from the **Send to Receiver(s)** list box.
7. Enter a descriptive ID for the package from the **With Identifier** text field. If you do not specify a descriptive ID for the delivery, the system will deliver the package with the default identifier (in this example the default identifier is "delivery1").
8. If you wish to have this package supersede all other deliveries currently in process, check the **Immediate Priority** check box.

   Immediate priority sets the package delivery to the highest priority. When ***deliver immediate*** is used, the package will be opened and executed before all other packages in the target's queue.

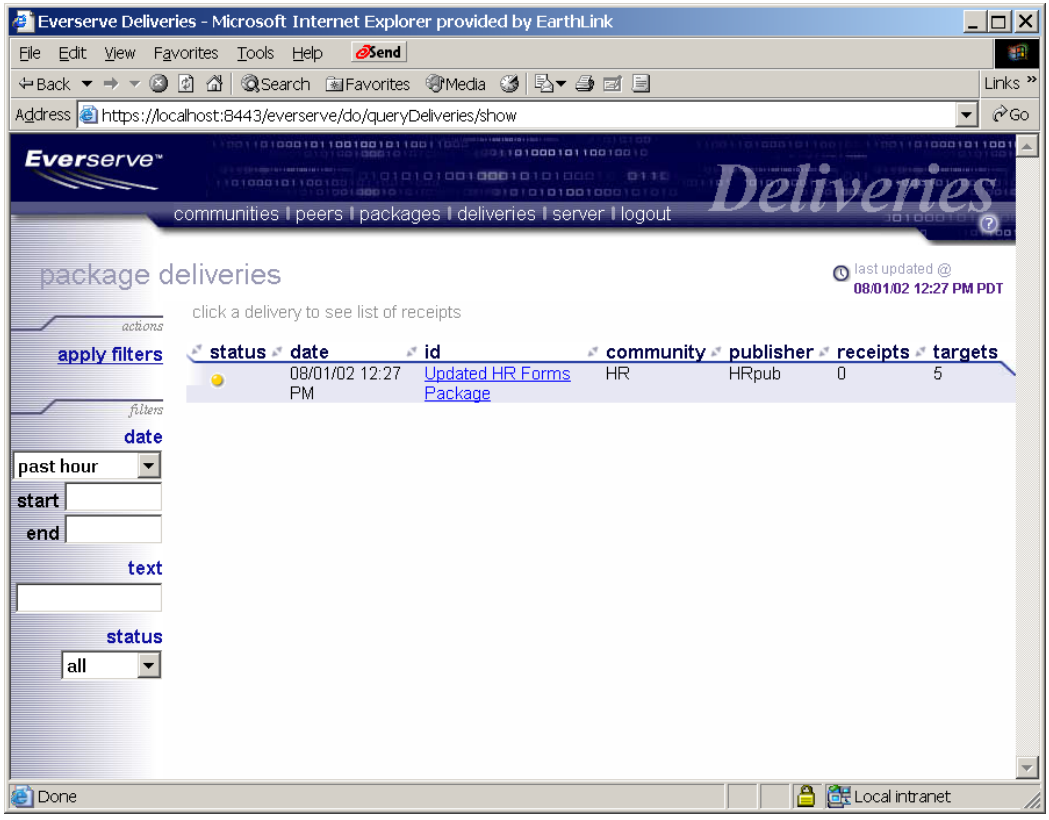
The following example illustrates specifying the package specification "*HRForms2002.xml*" for delivery to the community "*HR*," by the publisher "*HRpub*," sent to the receiver "*Relay 1*," using a package identifier of "*Updated HR Forms Package*:"

.



9. Click **Deliver** to send the package.
   The system sends the package to the receiver(s) specified.

The system displays the Deliveries page listing the deliveries made to the community or selected peers.



Status indicators are color coded to indicate the following delivery conditions:

Green Light:	All expected receipts have been received by the publisher and all packages were successfully delivered and executed on the target system.
Yellow Light:	One or more packages are pending; all receipts have not been received by the publisher.
Red Light:	One or more targets failed to execute the package (or an element in the package).

Viewing Return Receipts

Each target that receives and opens a package sends a return receipt back to the publisher that originated the delivery. The return receipt contains details (stdout) on the actions performed on the targets, any errors (stderr) experienced during delivery and/or execution of the package, and shows the package specification that was sent in the delivery.

To View a Return Receipt:

1. From the Deliveries page, set the delivery filters as desired and click **Apply Filters**.

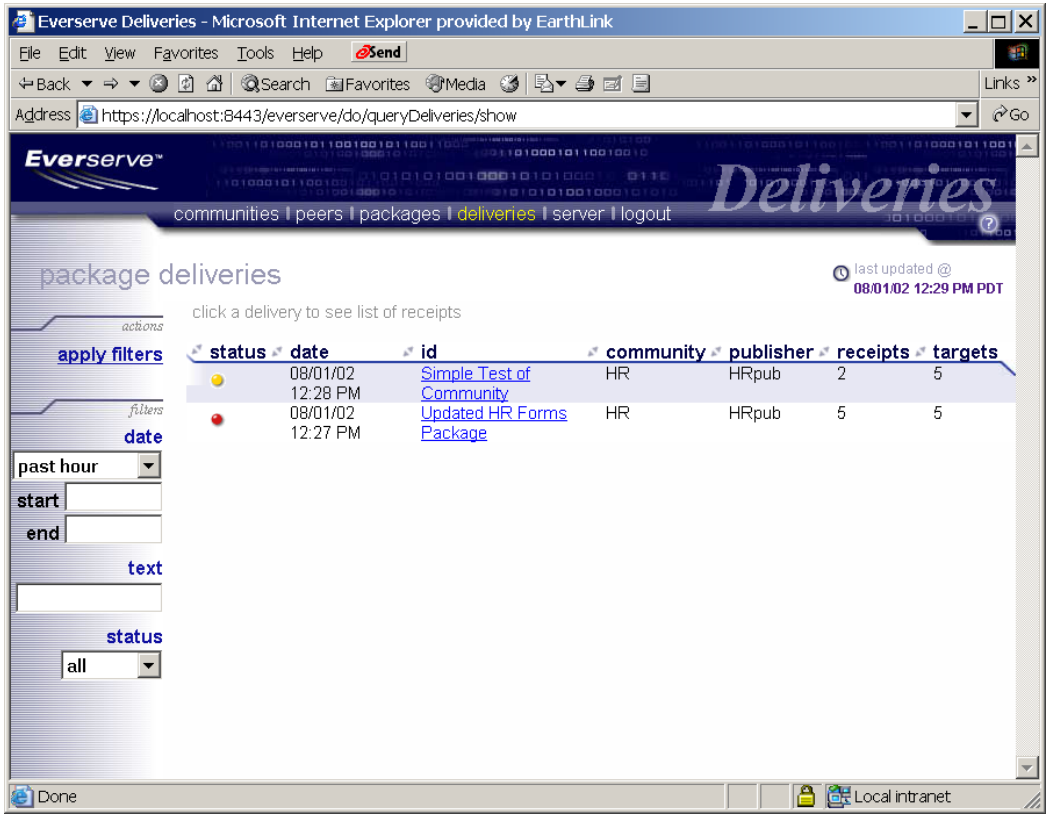
Page Filters

The following filters are available to view receipts:

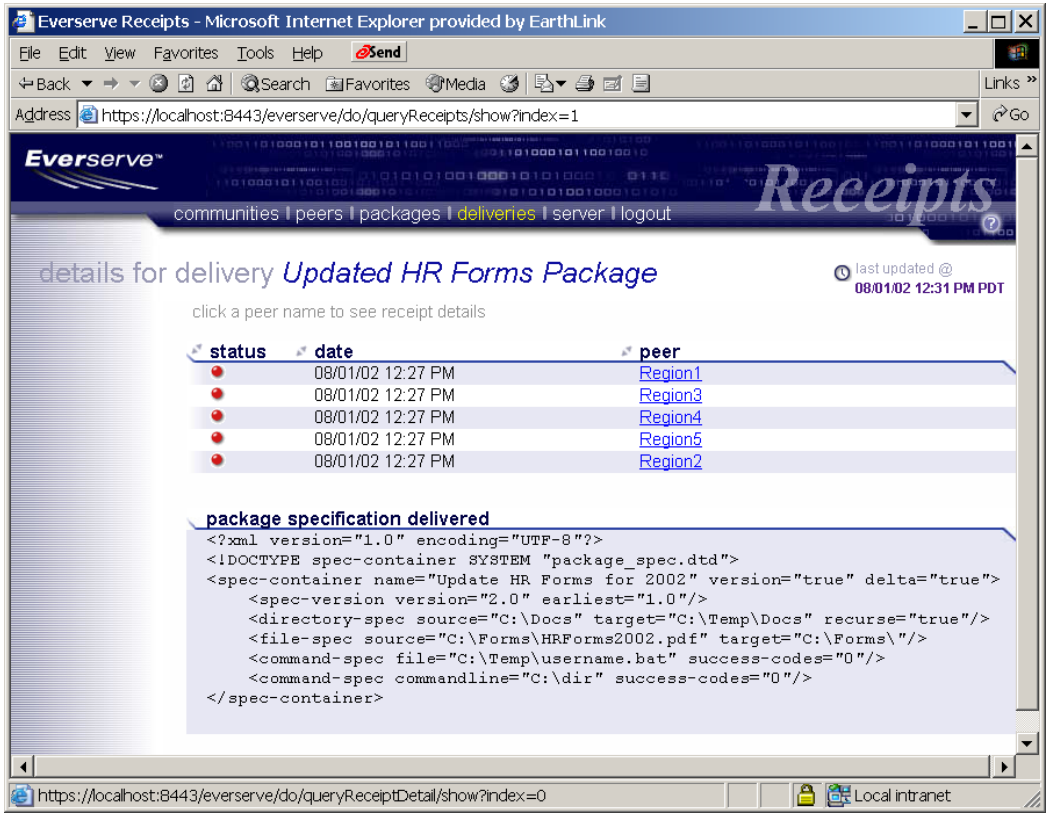
Filter Type	Description
Predefined Time/Date Range	View receipts using one of the predefined date/time ranges provided. Select the date range in which to view package receipts delivered to the device from the **Date** pull-down list. Date range options are as follows: • Past hour • Past 6 hours • Past 12 hours • Past day (24 hours) • Past 3 days • Past week • Past month • Specified (used in conjunction with Start and End date range)
Start and End Date Range	Specify date ranges for package deliveries. 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. • 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 end date. • The start date must be less than the end date, regardless of whether the dates are explicit or defaulted.

Filter Type	Description
Text	Descriptive text that may be found in the ID field of a receipt for a package delivery. For example, entering the text string "HR Forms" would result in a listing of all receipts whose original package ID contained the words "HR" and "Forms."
Status	Type of return receipt to view. Status types are: **All:** View all receipts that match the criteria regardless of pass or fail status. **Pass:** View only those receipts that match the criteria and were successfully delivered and executed. **Fail:** View only those receipts that match the criteria and that were found to have errors during the delivery or execution.

2. Click a delivery ID you wish to view.
   The system displays the Receipts page with the following delivery information:

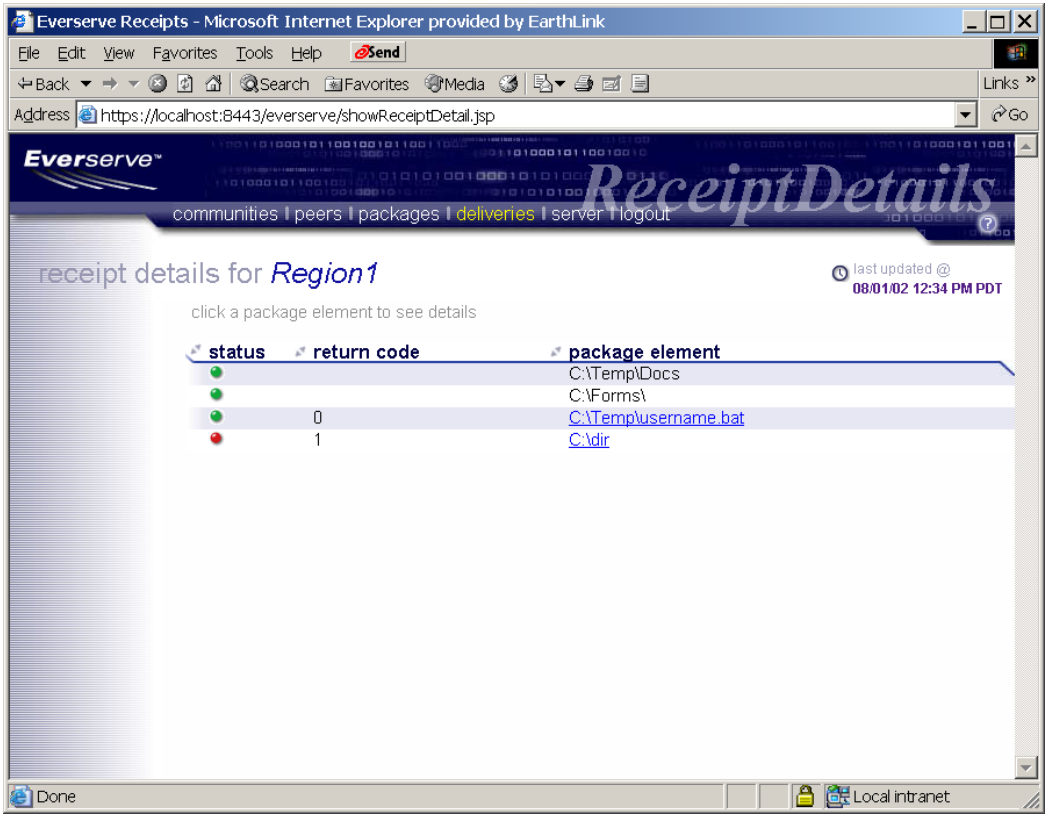


3. To view a return receipt for a specific delivery, click a delivery ID.
   The system displays the return receipt information for the package delivered to the peer:

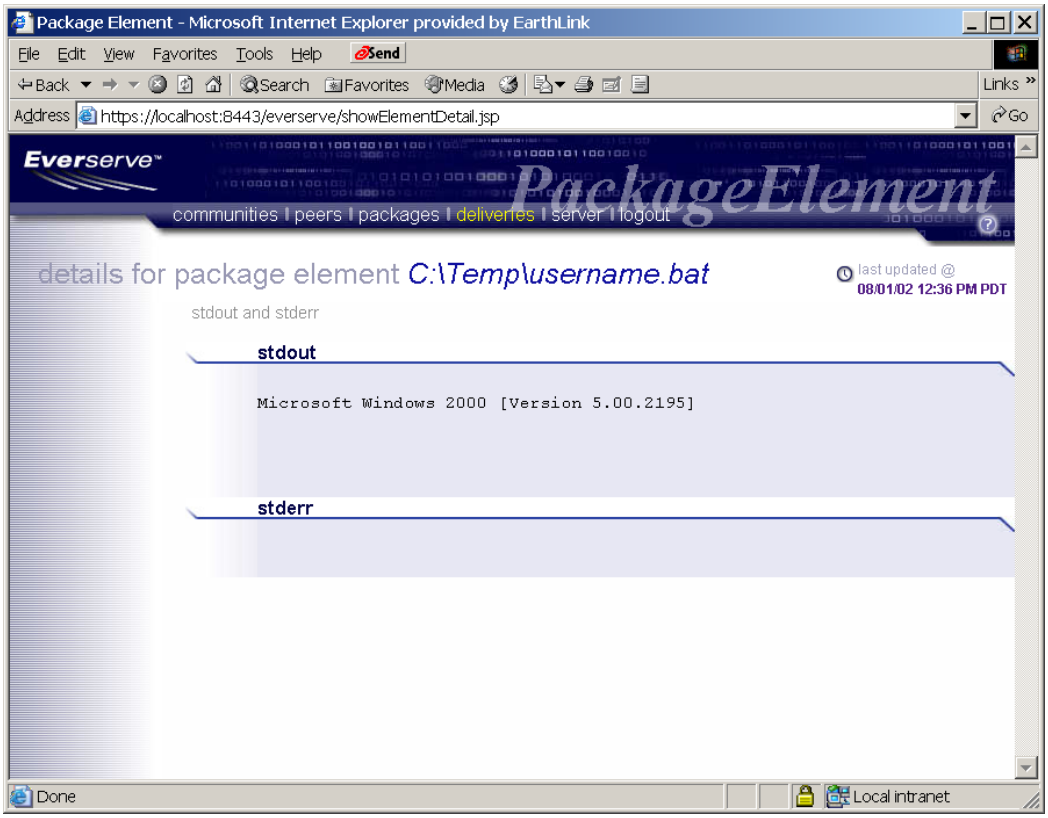


4. To view the details of a receipt for a peer, click the **Peer Name** link you wish to view.

The system displays the Receipt Details page showing the details of the delivery for the peer.

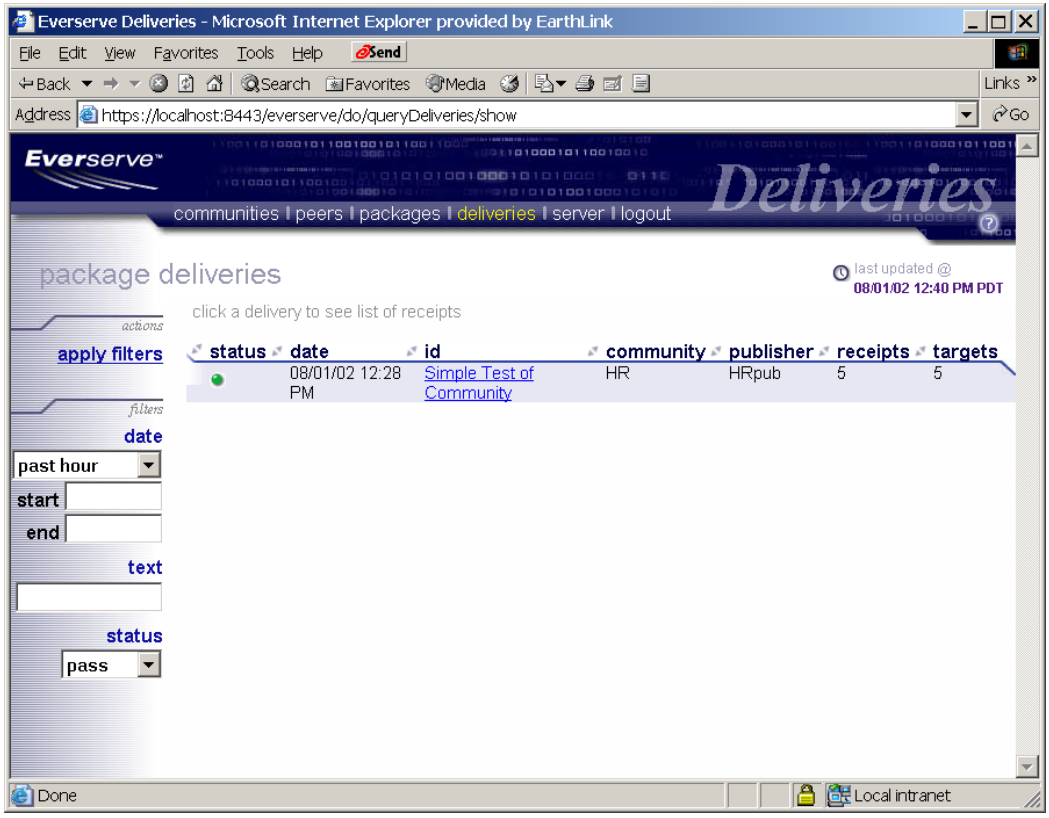


5. Click the **Package Element** link you wish to view.



The system displays the Package Element page showing the results of the specific command or action performed on the target.

The following example shows a receipt page for a delivery made. The status icon is green indicating no errors have been received. The number of receipts returned to the publisher is listed in the Receipts column:

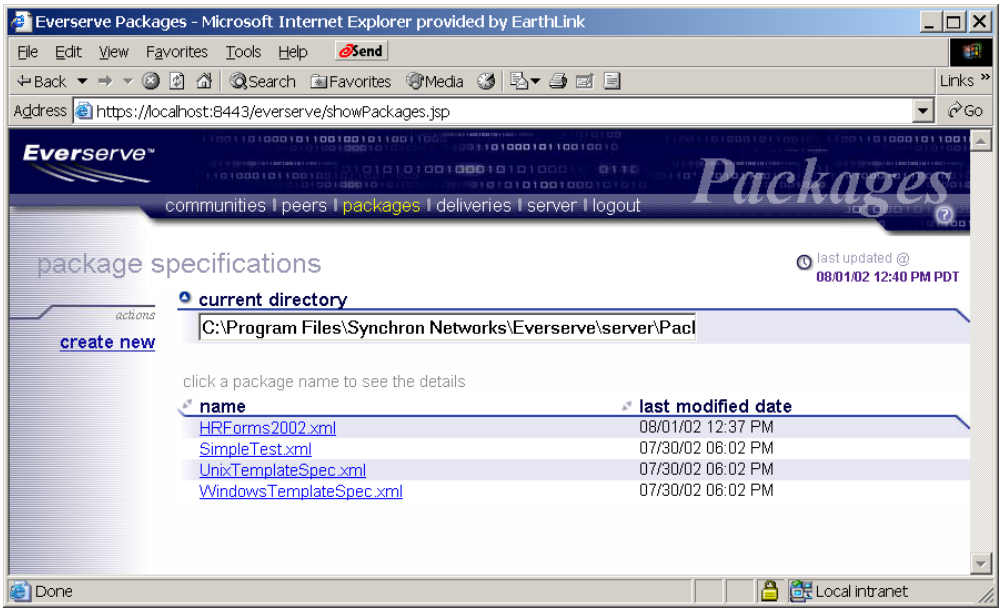


Deleting Package Specifications

There may be times when package specifications are created and delivered only once due to the nature of the package contents and recipient peers. There may also be occasions when a package specification is outdated and is no longer used. Everweb allows the deletion of package specifications, however, once deleted, the package specification can not be restored for reuse at a later time.

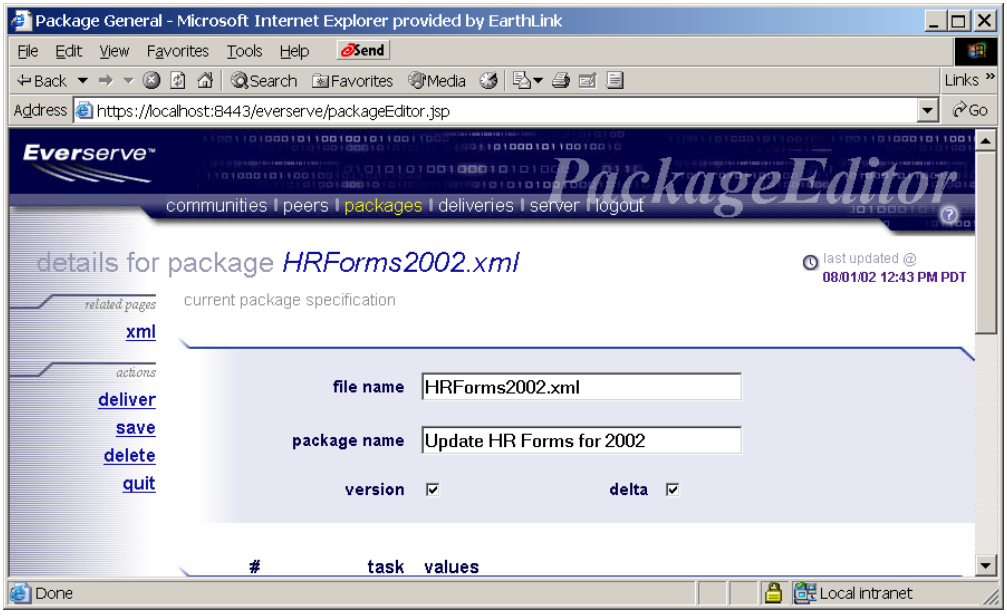
To Delete a Package Specification:

1. Click the **Packages** tab.
   The system displays the Packages page:



   The Packages page lists all packages specifications available from the current directory. To view packages in a different directory, specify the directory path in the **Current Directory** field.

2. Click the **Package Name** you wish to delete.
   The system displays the Package Editor page.



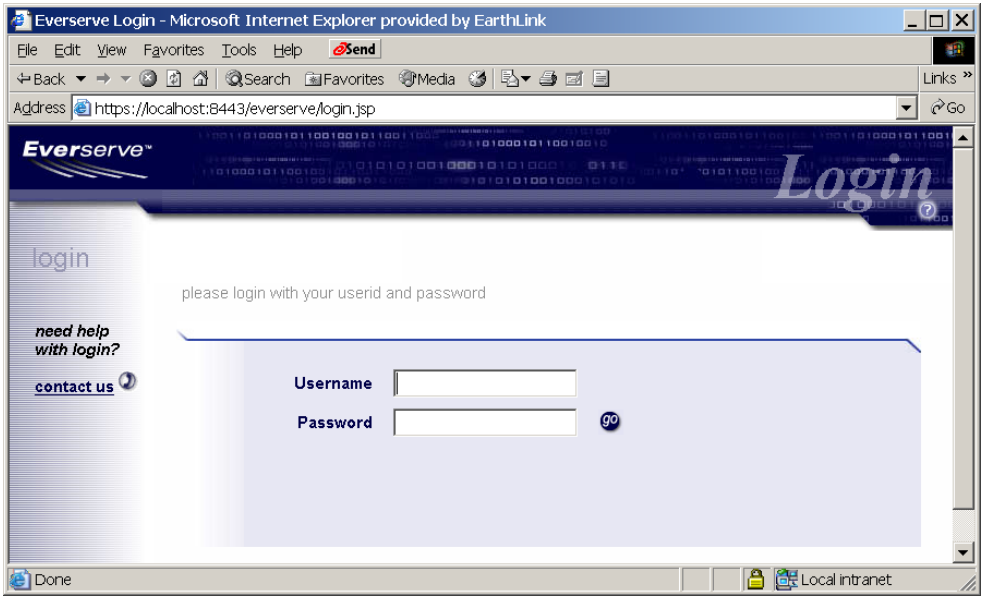
Note: Everweb does not display a confirmation dialog box for delete operations. Once the delete link is clicked the package specifications will be deleted.

3. Click **Delete**.

Logging Off the System

To logout of Everweb, click the **Logout** tab.

The system logs the current user out of Everweb and displays the user login screen.



Appendix A: Summary of Everserve Command Line Syntax

The commands listed in this appendix summarize the syntax used to manage communities, deploy and monitor package delivery, and to make inquires 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>]

Issuer	Command	Parameters
CM	create peer	{-p <peerName>} {-h <hostname>} [-d <description>]
CM	create seed	{-c <communityName>}
CM	delete community	{-c <communityName>}
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 <cummunityName>] [-p <peerName> [-i <ID>] [-f <filter>] [-s <startDate>] [-e <endDate>]
ALL	show deliverylog	[-a] [-c <cummunityName>] [-p <peerName> [-i <ID>] [-s <startDate>] [-e <endDate>

Issuer	Command	Parameters
ALL	show peers	[-c <communityName>] [-p <peerName>] [-h <hostName>] [-r <role>]
ALL	show processes	[-v] [-a] [-f <find>] [-s <startDate>] [-e <endDate>
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 A:Third Party Software

Use of the software products listed below is hereby acknowledged.

Bouncy Castle Crypto APIs

Copyright (c) 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.

Castor

Developed by the ExoLab Project (http://www.exolab.org/).

TWFreeTDS

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.

Jakarta-Struts, log4j, Servletapi - 3.2.2, Tomcat-3.2.2, Xerces-J 1.3.1

The Apache Software License, Version 1.1

This product includes software developed by the Apache Software Foundation (http://www.apache.org).

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 `\Synchron Networks\Everserve\server` 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

R

S

T

V

W