Peregrine

ServiceCenter

Java Client Installation and Configuration Guide

Release 5.1

PART NO: SCT-5.1-ENG-01008-00233

Peregrine SYSTEMS

If you need technical support for this product, or would like to request documentation for a product for which you are licensed, contact Peregrine Systems, Inc. Customer Support by email at support@peregrine.com.

If you have comments or suggestions about this documentation, contact Peregrine Systems, Inc. Technical Publications by email at doc_comments@peregrine.com.

This edition applies to version 5.1 of the licensed program.

Peregrine Systems, Inc.
Worldwide Corporate Headquarters
3611 Valley Centre Drive San Diego, CA 92130
Tel 800.638.5231 or 858.481.5000
Fax 858.481.1751
www.peregrine.com

Contents

Getting Started

The Peregrine Systems, Inc. Java client is an application that enables you to view ServiceCenter with a Web browser, or as a standalone application using a local Java Runtime Environment (JRE). This guide has instructions to install and configure the Java client.

The *Java Client Installation and Configuration Guide* has this information:

- *Getting Started* describes this guide and what you need to know. Provides product support information, lists client platform system requirements for the ServiceCenter Windows-based client or server, and how to contact Peregrine Systems, Inc. for customer support.
- *Before You Begin* on page 11 gives you a brief overview of the Java client, including pre-installation considerations and how the Java client installer works.
- *Installing and Starting the Java Client* on page 19 provides directions to install and start a Java client in a web browser or as a standalone application on all platforms.
- *Configuring the Client* on page 55 describes how to configure parameters and user preferences for the scjavalaunch.htm and sc.ini files.
- *Server Hub* on page 73 describes how to install and configure the server hub.
- *ServiceInfo Universal* on page 103 provides instructions to connect and configure your system for ServiceInfo Universal (SIU).

- *Troubleshooting* on page 107 provides information to troubleshoot the applet and standalone configurations.
- *Accessibility Specifications* on page 113 reviews the new features of Java client for usability, functionality, and integration with third-party disability software to ensure compliance with Section 508 of the Rehabilitation Act.

Knowledge Requirements

The instructions in this guide assume a working knowledge of Peregrine Systems ServiceCenter and the installation platform. You can find more information in the following guides:

- For information about a particular platform, see the appropriate platform documentation.
- For information about customizing your environment using parameters, see the *ServiceCenter Technical Reference* guide.
- Before you start the ServiceCenter server, see the *ServiceCenter User's Guide*.
- For administration and configuration information, see the *ServiceCenter System Administrator's Guide* or the *ServiceCenter Application Administration Guide*.
- For database configuration information, see the *ServiceCenter Database Management and Administration Guide*.
- For copies of the guides, download PDF versions from the CenterPoint web site using the Adobe Acrobat Reader, which is also available on the CenterPoint Web Site. For more information, see *Peregrine's CenterPoint Web Site* on page 9. You can also order printed copies of the documentation through your Peregrine Systems sales representative.

Examples

The sample windows and the examples included in this guide are for illustration only, and may differ from those at your site.

Contacting Customer Support

For more information and assistance with this new release or with ServiceCenter in general, contact Peregrine Systems' Customer Support.

Peregrine's CenterPoint Web Site

You can also find information about version compatibility, hardware and software requirements, and other configuration issues at Peregrine's Centerpoint web site: *http://support.peregrine.com*

1. Log in with your login ID and password.
2. Select **Go** for **CenterPoint**.
3. Select **ServiceCenter** from **My Products** at the top of the page for configuration and compatibility information.

Note: For information about local support offices, select **Whom Do I Call?** from **Contents** on the left side of the page to display the **Peregrine Worldwide Contact Information**.

Corporate Headquarters

Address:	Peregrine Systems, Inc. Attn: Customer Support 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+1 (858) 794-7428
Fax:	+1 (858) 480-3928

North America and South America

Telephone:	+1 (800) 960-9998 (US and Canada only, toll free) +1 (858) 794-7428 (Mexico, Central America, and South America)
Fax:	+1 (858) 480-3928
E-mail:	support@peregrine.com

Europe, Asia/Pacific, Africa

For information about local offices, see *Peregrine's CenterPoint Web Site*. You can also contact *Corporate Headquarters*.

Contacting Education Services

Training services are available for the full spectrum of Peregrine Products including ServiceCenter.

Current details of our training services are available through the following main contacts or at:

http://www.peregrine.com/education

Address:	Peregrine Systems, Inc. Attn: Education Services 3611 Valley Centre Drive San Diego, CA 92130
Telephone:	+1 (858) 794-5009
Fax:	+1 (858) 480-3928

CHAPTER 1 Before You Begin

The ServiceCenter Java client is a Java interface to ServiceCenter applications. The Java client supports the same functionality as the Windows client as well as enhanced features that are unique to the Java client and a Java environment.

Read this chapter for information about:

Java Client Features

The Java client has a number of features that make it more attractive than the Windows client in some instances. The Java client has:

- A tree structure navigational pane called ServiceCenter Explorer.
- Multiple Document Interface (MDI) support
- A Favorites toolbar
- Frequently used forms access
- Restore Forms on Startup option
- Multiple sessions capability
- Activity indicator in the status bar
- Message button in the status bar
- Windows save on exit
- Hyperlinks in a text field

Other Features

Because the client is a Java product, it can run wherever you create a Java environment. You can configure the client to run as a standalone application, or from a web server where the client can be downloaded remotely and runs in a web browser. The ServiceCenter administrator specifies the web server location and the Java client/server relationship during the Java client installation, or when the administrator configures the HTML files that launch the client in a browser.

The Java client supports the Microsoft Input Method Editor (IME). IME is a Microsoft feature that supports inline editing of languages with a large number of characters, such as Japanese.

The Java client uses TCP/IP sockets to communicate with a standard ServiceCenter server. You can install and start the Java client without any changes to an existing server environment.

Java Client Advantages and Limitations

There are many advantages to using the Java client. There is no client-side administration. Any connected Java client can download an upgrade as soon as the administrator makes it available on the server. You can upgrade the Java client to the latest version without upgrading the ServiceCenter server or applications. For example:

- The 3.0 SP3 Java client is compatible with all version 3.0 servers.
- The 4.0 Java client is compatible with all version 3.0 and 4.0 servers.
- The 5.*x* Java client is compatible with all version 3.0 and 4.0 servers.

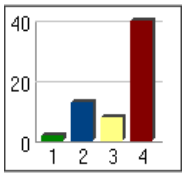
All ServiceCenter applications are supported without any modification or customization. The Java client interface can have the same look and feel as the Windows client interface, or you can use ServiceCenter Explorer (the tree structure) to navigate within the client.

You can print from the Java client, attach files, or use Object Linking and Embedding (OLE) support to create objects then link or embed them in the Java client. Because the number of tools and libraries that support Java is growing dynamically, it is likely the Java client will integrate seamlessly into a sophisticated Java environment.

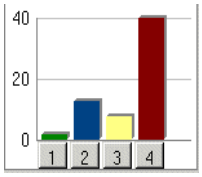
Limitations

There are a few limitations for Java client users:

- Chart buttons appear in the Windows client, but not in the Java client. The same functionality exists, however you must click the graphic bar in the Java client and click the buttons in the Windows client.



Java client without buttons



Windows client with buttons

- Because Java is still evolving, the same level of support is not available on all platforms.

The ServiceCenter Java client is based on Java classes included in the standard Java APIs, including Sun's Abstract Window Toolkit (AWT). The AWT is part of the Java Foundation Classes (JFC), which is the standard API for Java graphical user interfaces. All the classes needed by the ServiceCenter Java client, including the Swing components of JFC, are bundled and installed by this release.

Installer Features

The Java Installer eliminates downloading client and library files more than once, saving bandwidth and launching the client faster. The installer automatically detects client upgrades on the server and notifies you when you can upgrade the next time you run the client. It detects problems with the user's environment before the client is launched.

The installer provides easy cleanup of Java client-related files. When you remove the Java client, you also remove all related files. If you create new files in the installation folders, those folders are not deleted.

Requirements

You must specify the TCP/IP host name and service address of the ServiceCenter server during installation. The Java client must make a direct TCP/IP connection to the ServiceCenter server or connect through a server hub.

Installation Considerations

The ServiceCenter Java client can be installed and configured as either a standalone client or as a browser-based client. Before running the installation program, review the options for both types of installations and choose the one that suits your needs.

Browser-Based Clients

You can install a browser-based Java client on the same machine as the web server, or you can install it connect to a remote web server. Figure 1-1 on page 15 shows the relationship between a single ServiceCenter server and multiple Java clients.

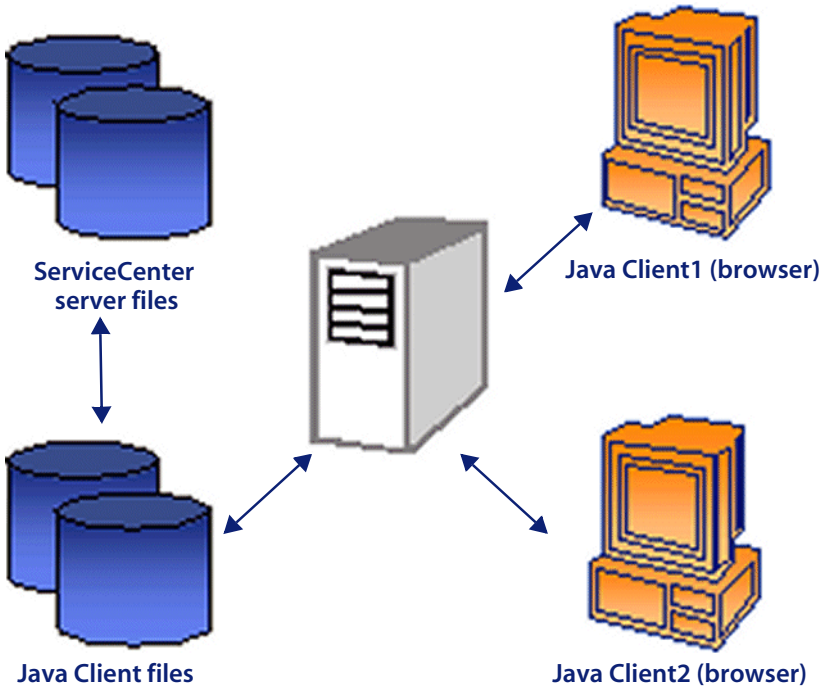


Figure 1-1: Java client/server relationship

The browser-based Java client can run from Internet Explorer and Netscape Navigator. To enable server access through the browser client, you must install the product on a web server or a network drive. When Java client users specify the URL for the **scjavalaunch.htm** file, they can launch the browser-based client. For information about browser version compatibility, see *Peregrine's CenterPoint Web Site* on page 9.

Launch files

The **scjavalaunch.htm** file that launches the Java client hides browser controls (the standard browser tool and menu bars) from the user and displays only the ServiceCenter controls. The **scjava.htm** file that launches the Java client displays both browser controls and ServiceCenter controls. Macintosh users can use the **scjavamac.htm** file to display both browser controls and ServiceCenter controls.

Some Unix systems cannot handle the signing mechanism used by the Java client. If you are a Java client user on one of those systems, you can connect through the scapplet.htm file. You can also use this connector if you do not have write permission to browser directories.

Standalone Clients

The Java client can be run as a standalone application when you have a supported JRE installed. Figure 1-2 shows how a standalone Java client has a local relationship with the ServiceCenter server.

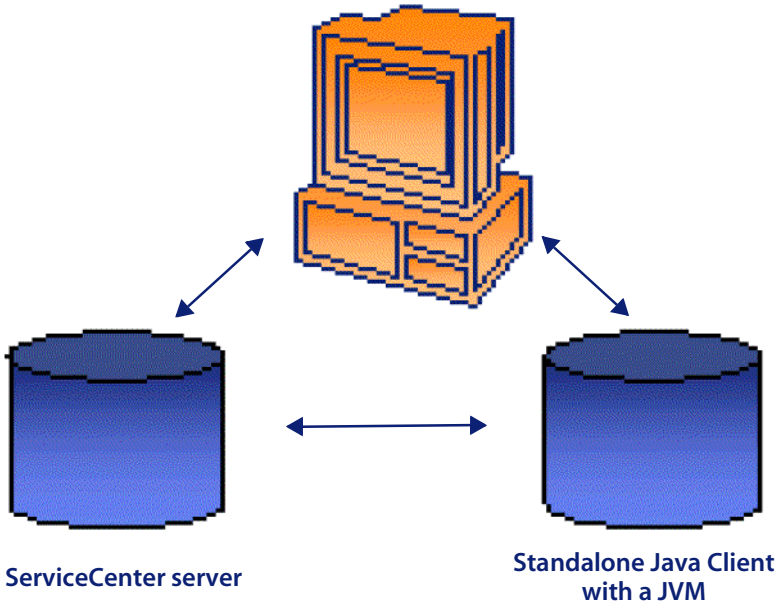


Figure 1-2: Standalone Java client/server relationship

The Java client may also be installed as a standalone application that runs with a Java Virtual Machine (JVM). For example, there are several Sun Java Runtime Environment (JRE) versions that create a compatible JVM environment. Other JREs, such as Microsoft JView, also supports the Java client. For compatibility information, see *Peregrine's CenterPoint Web Site* on page 9.

To create a standalone Java client, run the installation program on each client machine. You must specify the TCP/IP host name and service address of the ServiceCenter server during installation. If both the server and the standalone client reside on the same workstation or server, the TCP/IP host name and service address point to that machine.

Plug-in Support

The Java client can take advantage of Sun's Java Plug-in technology. Customers can download and install the Java Plug-in from Sun to run the Java client (or any other applet) using the latest Sun JRE. The ServiceCenter Java client will work with older JREs. Netscape 6.0 and later releases includes Java Plug-in support.

Windows XP does not include Java support, although some hardware manufacturers and resellers include an older JRE with their systems. Your system administrator can verify whether your system has installed Java support. You can download a JRE from the Sun web site. For more information, see *Installing the Java Runtime Environment* on page 51.

CHAPTER 2 Installing and Starting the Java Client

This chapter describes how to install the Java client on different platforms to run as an applet in a browser and as a standalone application. For more information about installing ServiceCenter on a Windows operating system, see the *Client/Server Installation Guide for Windows*. For more information about installing ServiceCenter on a Unix operating system, see the *Client/Server Installation Guide for Unix*.

Read these sections to learn more about installing the Java Client.

Windows Operating Systems

The Java client can run as a standalone client, a local browser-based client on a web server, or as a remote browser-based client

Installing a Standalone Java Client

You can install a standalone Java client by following the steps in this section, or you install a standalone Java client by following the steps in *Updating a Standalone Client* on page 30.

To install a standalone Java client in a typical installation:

1 Close all Windows applications, including ServiceCenter before you begin the installation. The Java client must access some shared .dll files. If you do not close other applications, you may encounter an error when you try to connect to or start the Java client.

The Java client standalone application requires a resident JRE. The ServiceCenter installation CD-ROM has a default JRE that you can install, you can use a Java Runtime Environment (JRE), version 1.2.2_008 or a later release, that you already installed. For complete information about current platform requirements and compatibility, see *Peregrine's CenterPoint Web Site* on page 9. When you determine which supported version you want to use, you can install this version directly from the Sun web site. For more information, see *Installing the Java Runtime Environment* on page 51.

2 If you want a standalone Java client with a ServiceCenter server on a single workstation, follow the steps in the *Client/Server Installation Guide for Windows* for a typical installation. When you finish the typical installation, proceed to the next chapter, *Configuring the Client* on page 55.

To install a standalone Java client in a custom installation:

1 Insert the ServiceCenter installation CD-ROM into the appropriate drive on your workstation or server. If you are installing on a system that has autorun enabled, the setup.exe file starts automatically. You can also choose one of these methods:

- Use Windows Explorer to navigate to the CD-ROM directory. Double-click autorun.exe.

- Start the ServiceCenter installation from the Windows command prompt. Type the following command:

  ```
  D:\>setup
  ```

  where D identifies the CD-ROM drive.

2 The Peregrine splash screen appears, as shown in Figure 2-1. Click **Install** to begin the installation.

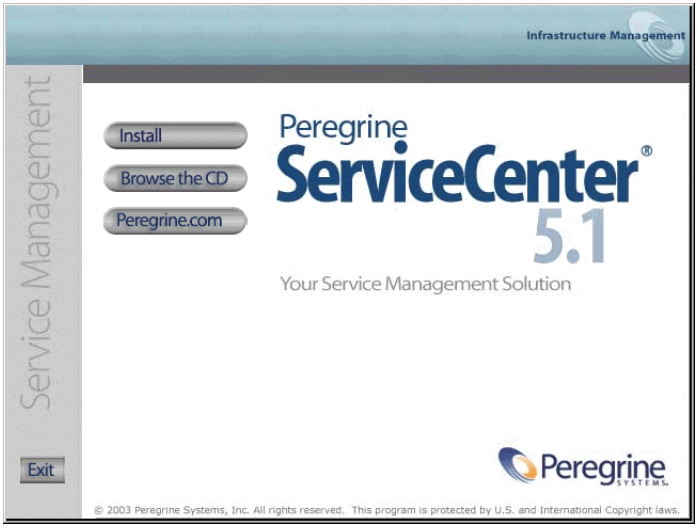


Figure 2-1: ServiceCenter splash screen

If the message shown in Figure 2-2 appears. Click **OK**. You can have multiple installations of the ServiceCenter server on the same workstation, and they can run concurrently. The installation program assumes a new directory each time you install a new instance of ServiceCenter. Earlier versions can continue to run as needed in separate directories; however, each instance must use a different port number.

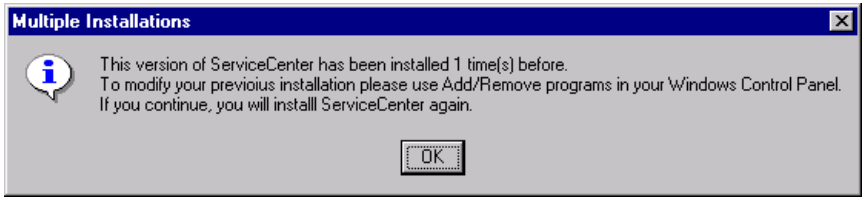


Figure 2-2: Multiple installations warning

3 InstallShield starts the setup wizard, shown in Figure 2-3. Click **Next**.

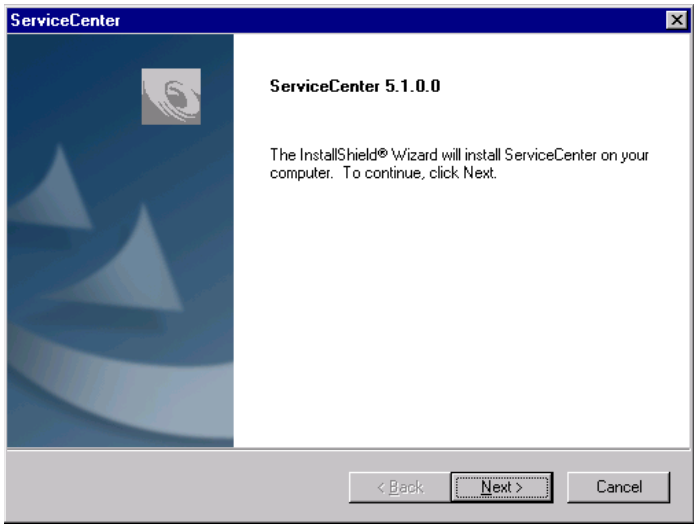


Figure 2-3: Setup wizard

4 When the Setup Type window appears, shown in Figure 2-4, select **Custom**.. Click **Next**.

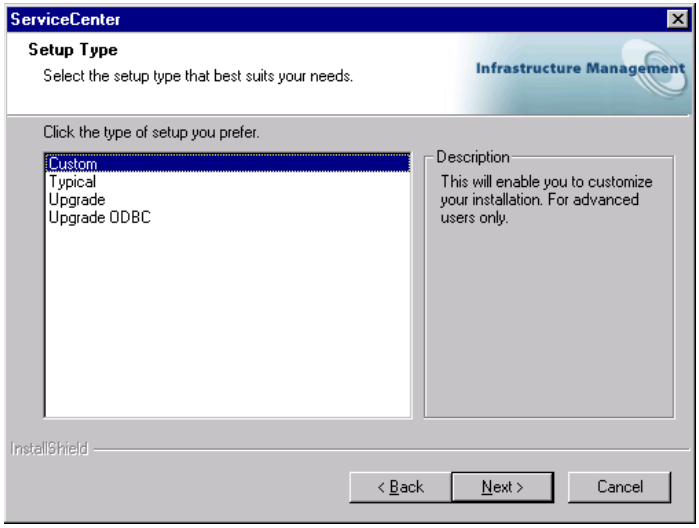


Figure 2-4: Setup Type window

Note: Click **Back** to return to a previous window to change your input. Click **Cancel** if you want to stop the installation altogether.

5 The Destination Location window appears. A typical installation creates a C:\Program Files\Peregrine\ServiceCenter folder. Figure 2-5 shows the default destination location. Click **Browse** to choose a different location. Click **Next**.

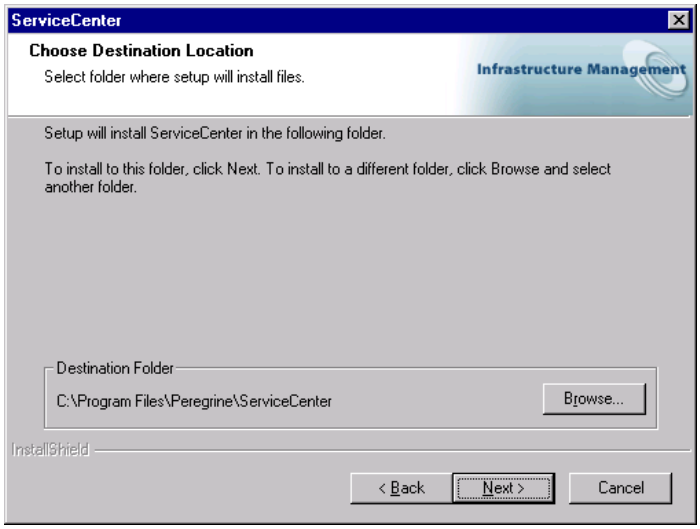


Figure 2-5: Choose Destination Location window

If this is a multiple instance of a Java client, the installation assigns a unique folder name, such as C:\Program Files\Peregrine\ServiceCenter2, or you can choose a custom location.

To install another type of Java client to an existing installation, you must modify that ServiceCenter installation through the Windows Control Panel. For more information, see the *ServiceCenter Client/Server Installation Guide for Windows*.

6 The Select Components window appears. Clear all check boxes except **Java Client**, as shown in Figure 2-6 on page 24. Click **Next**.

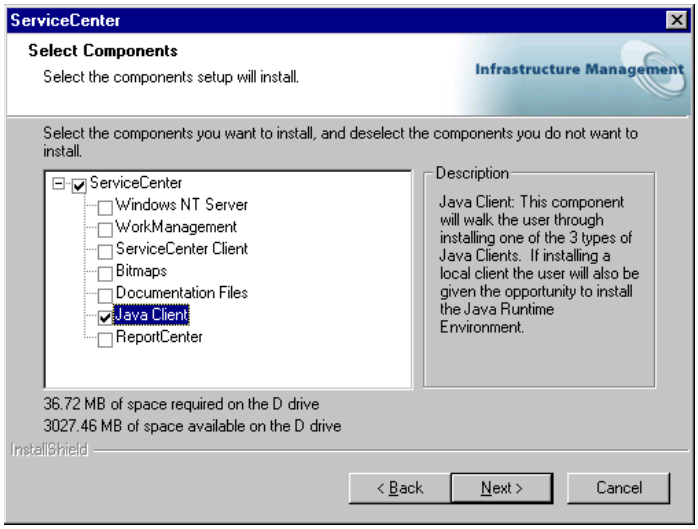


Figure 2-6: Select Components window for a Java client

7 Figure 2-7 shows the License and ReadMe Information. If you scroll through this information, you can learn more about licensing, accessing the release notes, Peregine's CenterPoint Web site, and customer support information. Click **Next**.

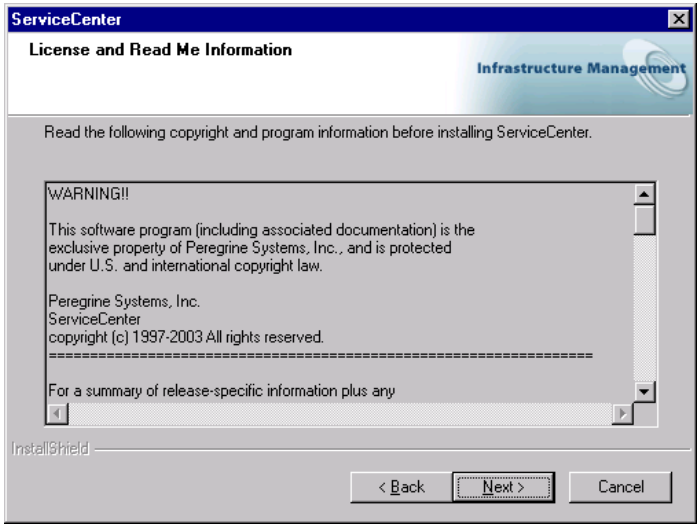


Figure 2-7: License and ReadMe Information window

8 Figure 2-8 shows the TCP/IP Server Information window. The installation automatically detects the assigned Service ID (port number) for the workstation or server. If no port number appears, contact your system administrator to troubleshoot the TCP/IP connection. Click **Next**. The installation validates the host name.

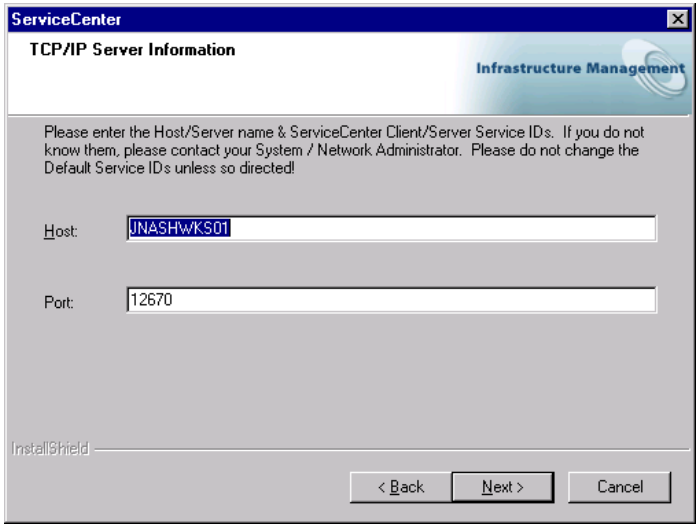


Figure 2-8: TCP/IP Server Information

Important: If the default port number is already assigned to another ServiceCenter server, you must choose a different port number to avoid conflict when you run multiple instances of ServiceCenter.

9 The Java Client Option window appears. Figure 2-9 shows the Standalone client selected. Click **Next**.

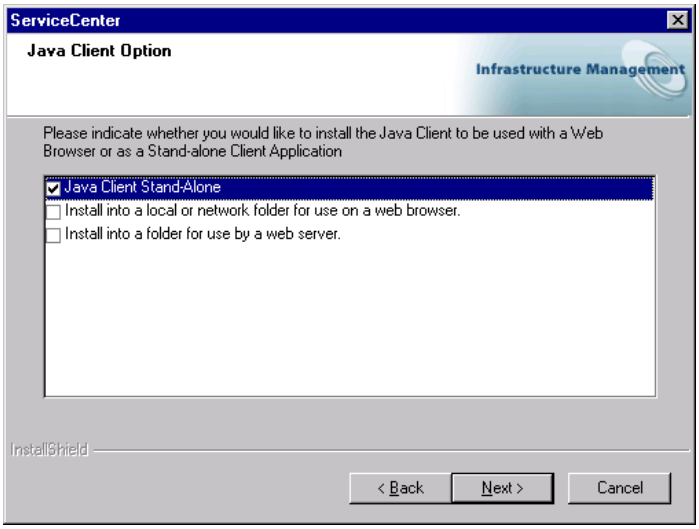


Figure 2-9: Java Client Option

10 Figure 2-10 shows the Java Runtime Environment window.

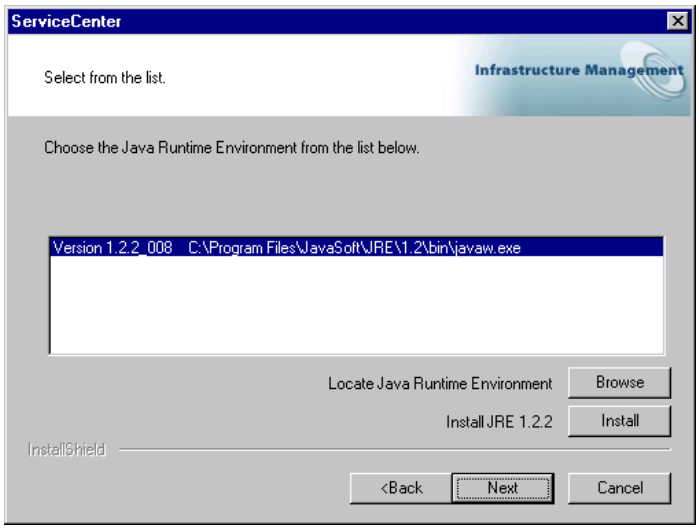


Figure 2-10: Select JRE window

The ServiceCenter Java client requires a JRE to run as a standalone application. The JRE creates a Java Virtual Machine (JVM) that this standalone application requires. If you have one or more JRE versions installed, select the one to be used with ServiceCenter from the list, or you can browse to the location of the desired JRE if it does not appear on the list.

If you do not have a JRE installed, click **Install** to use the version on the installation CD-ROM. The JRE shipped with this release is Sun's Java 2 Runtime, version 1.2.2_008. If installed, this version becomes the default JRE for the system. If you have a different version already installed, verify that the version is recommended for ServiceCenter on *Peregrine's CenterPoint Web Site* before you select it.

Figure 2-11 shows a selected JRE. Whether you choose an existing JRE, or you installed the version shipped with ServiceCenter, click **Next**.

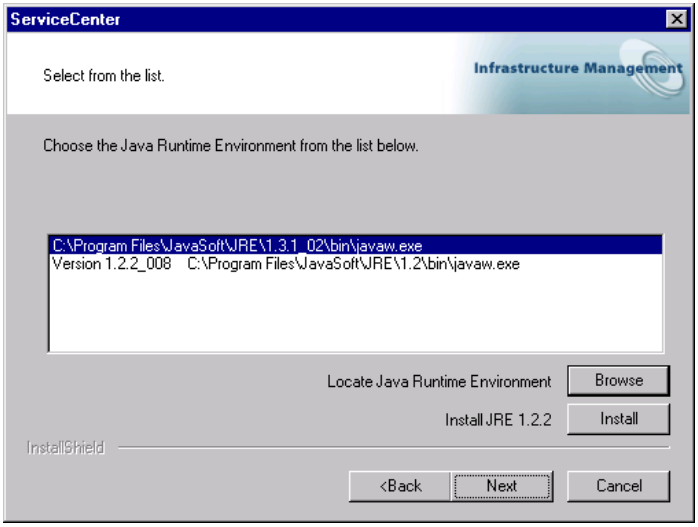


Figure 2-11: Java Runtime Environment window

11 The Select Program Folder appears, as shown in Figure 2-12 on page 28. The installation program creates a new ServiceCenter program folder or allows you to type a different program folder name. Click **Next**.

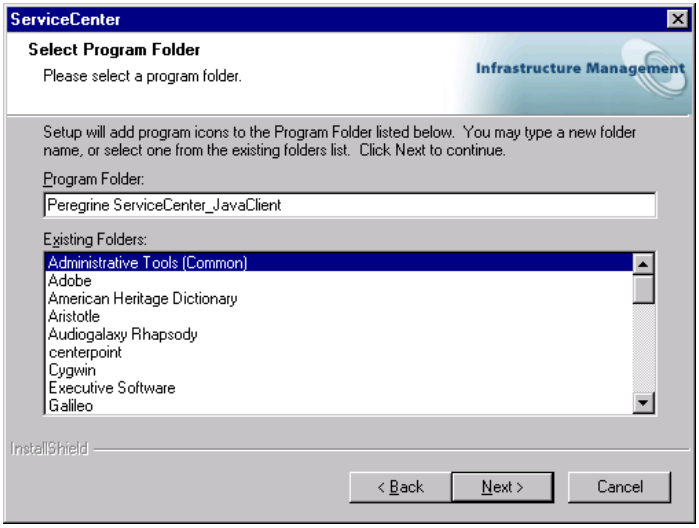


Figure 2-12: Select Program Folder

12 The installation program has enough information to start copying files into the designated program directory. Figure 2-13 shows the summary of settings that you requested during the setup process. Click **Next**.

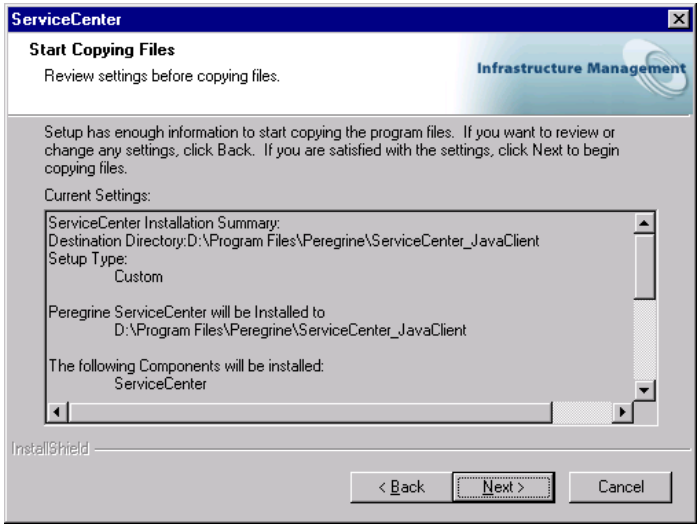


Figure 2-13: Start Copying Files window

13 The installation begins copying the selected files, as shown in Figure 2-14. You can stop the installation if you click **Cancel**.

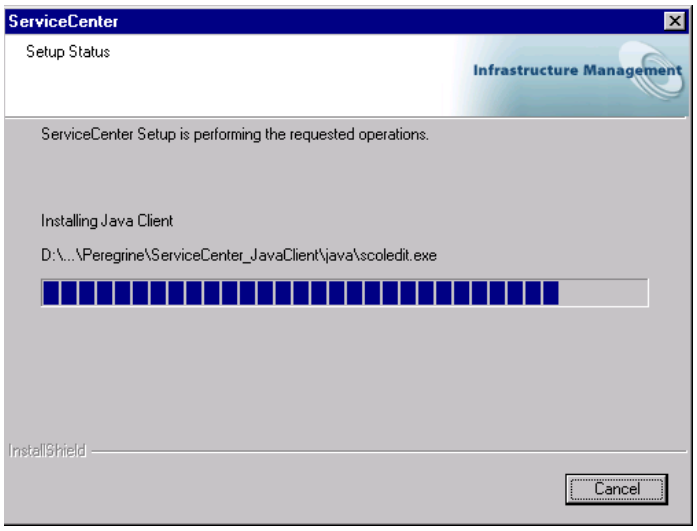


Figure 2-14: Setup Status window

14 When the Java Client installation finishes, the window shown in Figure 2-15 appears.

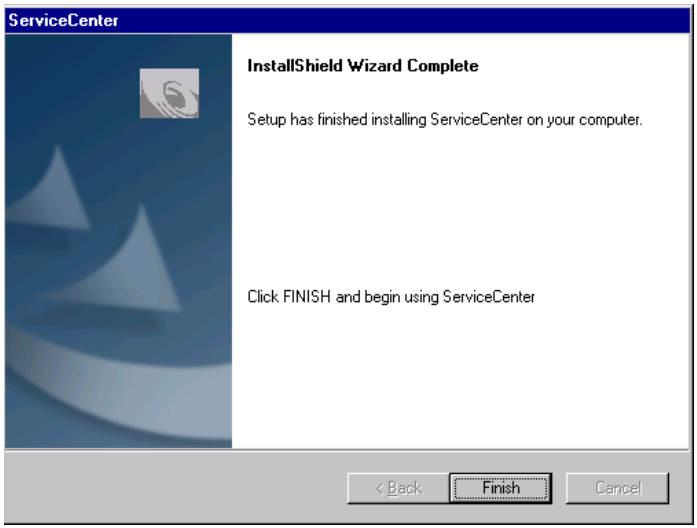


Figure 2-15: installShield Wizard Complete

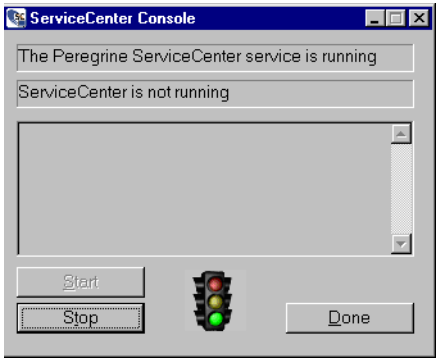
Running a Standalone Client

The ServiceCenter server must be running before you can start the Java client.

1 To verify the status of the server, from the Windows **Start** menu on the client workstation, select **Programs** > **Peregrine ServiceCenter** > **ServiceCenter Console.**

2 The ServiceCenter console shows a running server with a green light. If the light is red or yellow, click **Start** and wait for the green light.

3 From the Windows **Start** menu, select **Programs** > **Peregrine ServiceCenter** > **Java Client**. The ServiceCenter login window appears.



Updating a Standalone Client

You can update or install a browser-based standalone Java client instead of using the ServiceCenter 5.1 installation CD-ROM. Before you begin, you must have ServiceCenter 5.1 installed on a local or remote web server. The browser-based standalone client communicates with the server-side installation to obtain updates.

Follow these steps to update or install:

1 Create a copy of the scjava.htm file.

2 Open the copy of the ..\java\scjava.htm file with a text editor and add the following parameters

- <param name="InstallType" value="Standalone">
- <param name="SCJ_Home" value=SERVICE_CENTER_JAVA_HOME>
- <param name="JRE_Home" value=JAVA_HOME>

where SERVICE_CENTER_JAVA_HOME should be a directory on the client system. For example, c:/scjavaclient and JAVA_HOME are locations where the jre resides on the client system. For example:

c:/program files/javasoft/jdk1.3.1

3 Change the ImagePath parameter value in the .htm file to SERVICE_CENTER_JAVA_HOME/bitmaps.

4 Open the modified scjava.htm file with a browser. When the file opens, one of the following will happen:

- The standalone Java client will install in the specified directory in the SCJ_Home.
- The existing standalone Java client installed in the specified directory will update if it is a different version.
- The existing standalone Java client installed in the specified directory will launch if it is the same version.

Installing a Browser-Based Java Client

When the Java client runs as a browser application, the ServiceCenter server software can reside on a local or remote server. The client can be:

- A browser-based client that is local to the web server running ServiceCenter.
- A browser-based client that is remote to the web server running ServiceCenter.

  Before you begin using this Java client, you need the Uniform Resource Locator (URL) of the server where the Java client launch file scjavalaunch.htm resides.

To validate browser compatibility, see *Peregrine's CenterPoint Web Site* on page 9.

A local browser-based client

This client communicates only with a local server; therefore, no URL is required.

To install a local browser-based Java client:

1 Follow step 1 through step 8, beginning on page 20 for a standalone Java client.

2 The Java Client Option window appears. Figure 2-16 on page 32 shows the web browser client selected. Click **Next**.

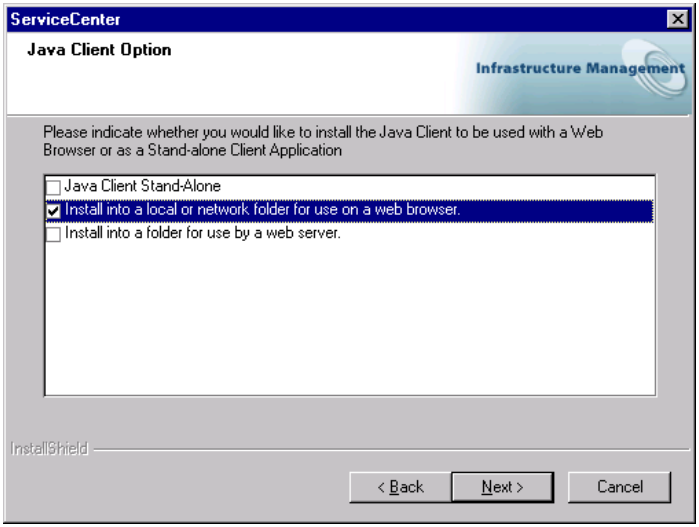


Figure 2-16: Browser Client option

3 Return to the Java client installation beginning with step 11 on page 27. You can skip step 10. A browser-based Java client does not require a JRE.

To validate browser compatibility, see *Peregrine's CenterPoint Web Site* on page 9.

A remote browser-based client

This client communicates only with a remote server. You must specify a URL to make the remote connection.

To install a remote browser-based Java client:

1 Follow step 1 through step 8, beginning on page 20 for a standalone Java client.

2 The Java Client Option window appears. Figure 2-17 on page 33 shows the web browser client selected. Click **Next**.

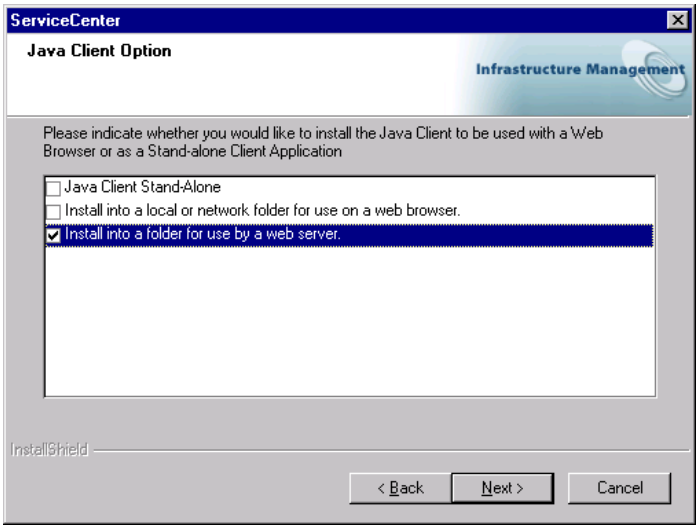


Figure 2-17: Web Client option

3 The Java Client window appears. Type the URL for the web server that hosts the ServiceCenter server. Do not include the prefix http:// when you type the URL. Follow the example shown in Figure 2-18. Click **Next**.

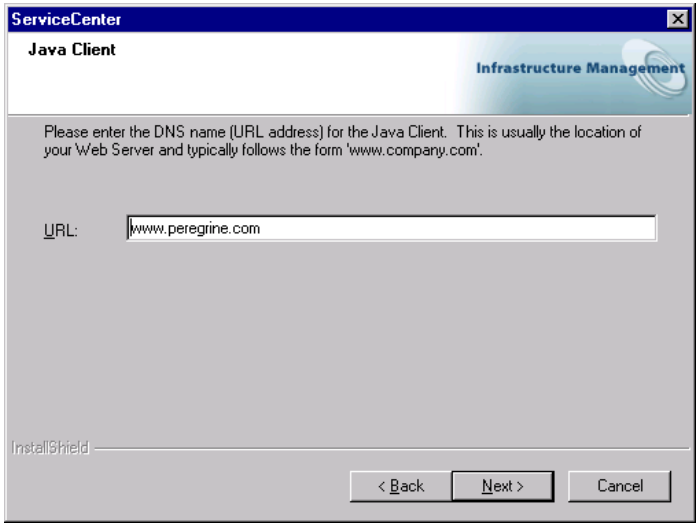


Figure 2-18: Java Client URL

4 Return to the Java client installation beginning with step 11 on page 27. You can skip step 10. A browser-based Java client does not require a JRE.

Running a Browser-Based Java Client

When you run the ServiceCenter Java client from a web browser, the Java client installer downloads a small applet that enables the client to run when the browser points to the correct server URL.

To run a browser-based Java client:

1 From the **Windows Start** menu, select **Programs**, choose **Peregrine ServiceCenter**, and click the **Java Client** with the browser icon. The Windows Java client uses the ServiceCenter icon. When execution begins, the Java client collects environment information to verify that the client is running on a supported platform and browser.

Figure 2-19 shows a security warning that appears. Click **Yes** for a single session, or click **Always** to bypass this message in the future.

Figure 2-19: Security warning

The Java client checks the local disk for an existing version of the downloaded Java client. If a current Java client already exists, the installation program launches the client and initiates a ServiceCenter session.

2 If no Java client is found, or if the client found is older than the client version on the server, one of two windows appears. When you choose the type of download, the installation program downloads the client applet files. When the download is complete, the client starts running. Figure 2-20 shows the available options. Click **Upgrade** to download files from the server, or **Continue** to use the available files on the workstation.

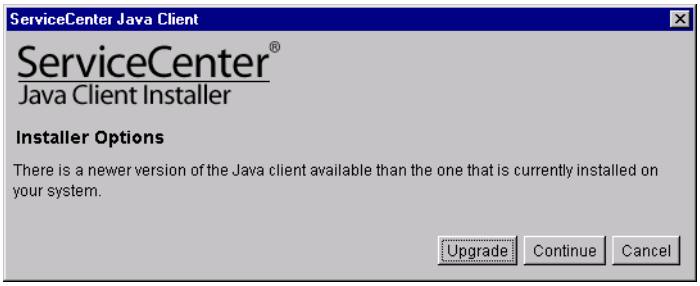


Figure 2-20: Installer options

Note: The Java client installation begins through the Java class com.peregrine.sc.installer.ClientInstaller. The Unix client applet begins through com.peregrine.sc.client.ClientApplet. The standalone client begins through com.peregrine.sc.client.ClientApplication.

Downloaded files

The Java client installer downloads certain files and installs them into a directory that is in the browser's Java class path. Netscape users require only the image files.

File Name	Description
Java client applet	Contains the ServiceCenter client code (about 1600K).
Java client images	Contains bitmaps and images used by the client (about 600K).
Swing Library	Contains the Java Swing Library, which is a set of GUI APIs. In the future, some browsers may include this library, eliminating the need for this download (about 2.4MB)
scoicon.exe	Contains display images with attachments (about 100K).

Bitmap files

The Java client installation program normally downloads and caches image files. However, you can specify an **imagepath** parameter that defines the location of files that are not already in the cache directory. Use this technique to add bitmap files to ServiceCenter applications and make them available to Java client users.

Choose one of these methods to add bitmaps:

- Add the bitmaps to the **bitmaps.zip** file, and update the **scjversions.properties** file in the web server directory.
- Add the bitmaps to a web server directory, and include that path in the **imagepath** parameter.

Htm files

To access the Java client through a browser, you must connect to one of the files in the following table using a URL and the directory location of the file. For example:

http://*yourserver*/java/scjavalaunch.htm

where *yourserver* is the name of your ServiceCenter server.

For more information about this type of configuration, see *Browser-Based Clients* on page 14.

Htm File	Function
scapplet.htm	Used primarily by some Unix systems that have difficulty with the signing mechanism used by the Java client running in a browser. The scapplet.htm file uses a different signing mechanism. Access this file directly, not through the scjavalaunch.htm file. The scapplet.htm file automatically uses the latest version of the client found on the server. This file also works on Windows systems.
scjava.htm	Use this file on any platform (except Macintosh) that supports the standard browser signing mechanism, and will automatically download new client files if it finds those files on the server. Access this file directly, not through scjavalaunch.htm, to display the client in a browser with the browser controls present. You must specify a URL to the scjava.htm file for users to connect to the web server.

Htm File	Function
scjava13plugin.htm	This file launches ServiceCenter with JRE 1.3 if you have it installed. If you connect to scjava13plugin.htm and the plug-in is not installed, the Java client prompts the user to download and install the a supported JRE from Sun.
scjavalaunch.htm	Use this file for Windows or Macintosh systems. This file is the primary connection file. It contains a Java script that: • Determines whether the platform is Windows or Macintosh • Launches the appropriate .htm file in a browser with the browser controls hidden.
scmac.htm	Use this file with Macintosh systems. Point your browser to the scmac.htm file located on an accessible Windows or Unix web server. If you need modifications, such as configuring the server hub, you must make them in the scjavamac.htm file. Local installation only: Macintosh supports the standalone version of the Java client. Access this file directly to start the client in a browser with the browser controls present.

Launching a Browser-Based Client

The ServiceCenter server must be running locally or on the remote web server before you can start the Java client successfully.

To run a Java client in a browser:

▶ Do one of the following:

- From the Windows Start menu, select **Programs** > **Peregrine ServiceCenter** > **Java Client**.
- From your browser, type a URL to the scjavalaunch.htm or scjava.htm file to connect to the web server. For example:

  http://*yourserver*/java/scjavalaunch.htm.

  http://*yourserver*/java/scjava.htm.

The scjavalaunch.htm (or scjava.htm) file opens the scjavamac.htm file, which appears in your client browser. When you connect, the ServiceCenter login screen appears in the Web browser.

Note: The ServiceCenter server and the web server the user connects to must both be started.

Changing the Java Client Heap Size

If you run a Sun JRE, follow these steps.

1. From the Windows Start menu, right click **Programs** > **Peregrine ServiceCenter** > **Java Client**.
2. Select **Properties**.
3. Insert `-Xms64 -Xmx64` in the Target path before the class path.

Unix Operating Systems

The main installation CD-ROM contains the Java client Unix installation. You can install and configure the ServiceCenter Java client as a standalone client or as a browser-based client. Before running the installation program, review the options for both types of installation and choose the one that suits your needs. For more information, see *Installation Considerations* on page 14.

Note: The Java client cannot connect to a port number greater than 65535.

Warning: Do not install the Java client as root (superuser).

Prior to installation:

1. Insert the ServiceCenter CD in the drive.
2. Ensure that you mount the CD-ROM drive.
3. Create a directory under the web server document root (if you plan to make the Java client available as a web URL), or create a directory for it elsewhere on your system. If you create this directory as root, give ownership and permissions to the ServiceCenter administrative user. You must have the appropriate permissions to create directories. (The installation script attempts to create directories that you specify if they do not exist.)
4. Ensure that the directory you create for the Java client installation is in the path of the ServiceCenter owner.
5. Change directories to the CD-ROM drive.

To install the ServiceCenter Java client:

1. Change directories to the Unix directory on the installation CD-ROM.
2. Run the executable installation script (`install.sh`).

3 Type the Java client installation root directory, which is the directory that will contain the Java client files. For example, if you type ServiceCenter, the Java client files will reside in the /ServiceCenter/java directory.

Note: The system validates any directory name that you specify. If you specify an invalid directory, the installation generates an error message. If the installation cannot validate the directory name after three attempts, the installation script exits and generates an error message.

4 Choose the product to install:

- ServiceCenter (Includes Java Client)
- Java Client

Select **option 2, the Java Client**.

5 Follow the prompts, using the following information for a standalone or browser-based client installation:

- **Is this a browser-based client install?**
  Type **n** for a standalone Java client installation.
  - **Service variable:** Type the ServiceCenter service ID number that the Java client connects to, such as `12670`. Enter numeric values only. If ServiceCenter is running as a named service (as defined in your system `etc/services` file) do not specify the service name itself. Type the port number assigned to the named service.
- **Is this a browser-based client install?**
  Type **y** for a standalone Java client installation.
  - **Codebase variable** - not used by the standalone installation. Type the URL for the Java client installation directory. For example, if you install the client in a directory named `java` under your company web server's document root, the codebase variable would be `www.mycompany.com/java`. Do not type **http://**.
  - **Service variable** - type the ServiceCenter service ID number that the Java client connects to, such as `12670`. Enter numeric values only. If ServiceCenter is running as a named service (as defined in your system `etc/services` file) do not specify the service name itself. Type the port number assigned to the named service.

Running a Standalone Client

You can run the Java client as a standalone application after you install ServiceCenter on a system that the Java client can connect to. The JRE must appear in the user's path before you can run the Java client as a standalone application on a Unix workstation.

To run the ServiceCenter Java client:

1. Change directories to the Java client installation directory.
2. Change directories to /Run.
3. Type **scjava** at the command line. The ServiceCenter Java client starts as an application.

Command Line Parameters

The **scjava** script has several parameters that you can specify; however, be sure you understand the settings before you use the parameter. Incorrect parameter settings cause the Java client to fail.

Syntax

At the command line, type:

```
scjava -parameter argument
```

Type the command, the parameter, and the argument to change the default.

Parameter	Description and Examples
-- <-AppArgs>	Specifies additional arguments that the scjava launch script does not interpret. The launch script passes these arguments to the Java application. If you use this parameter, it should be the last one specified on the command line. Use this method to pass any valid Java client parameter. For example, to pass an application argument, type: scjava -- -huburl http://webserver To pass a standard JRE location argument followed by an application argument: scjava -java /bin/jre/ -- -huburl http://webserver
args	Specifies arguments to pass to the JVM. Some common examples are -mx and -ms to control heap allocations. To review the supported options and valid syntax, see your JVM documentation. If you specify multiple sets of JVM arguments or a JVM argument contain spaces, enclose them in quotation marks.
java	Specifies the path to the JVM.
host	Specifies the name of the host workstation or server running the ServiceCenter services. If you omit this information, the launch script parses the installed HTML file (scjavalaunch.htm) for the correct information.
-service	Specifies the service number (port) used by the host workstation or server. This port number is typically 12670. If you omit this information, the launch script parses the installed HTML file (scjavalaunch.htm) for the correct information.
-images	Specifies the URL for the ServiceCenter images. If you omit this information, the launch script assumes its own location as the default location of the images.

To view the list of parameters within the script, type the following at the command line:

```
scjava -help
```

If you are a system administrator, you can edit the scjava scrip. Please read the comments in the file carefully before you make any changes. It is a good practice to save a backup of the file before you save new changes.

Running a Browser-Based Client

To enable users to run the Java client in a browser on Unix systems, system administrators must do one of the following:

- Method 1: Perform the initial start of the Java client as root.
- Method 2: Change the permissions on the Netscape directory to provide full permissions for users.

Note: If the users have Windows or Macintosh operating systems to connect to a Unix web server, you can ignore these steps.

Method 1:

1. Install the Java client as a regular user.
2. Start the Java client as a root user in a browser using the scjavalaunch.htm file. The client will automatically upgrade. Thereafter, users can start the Java client in a browser without root access. Users will need root access only to upgrade the client.

Method 2:

1. Install the Java client as a regular user.
2. Recursively change permissions on the java/classes directory under the Netscape home directory to grant full access (read, write, execute) for a single user or set of users.

Launching the Client

Specify a URL to the scjavalaunch.htm or scjava.htm file to connect to the web server. For example:

http://yourserver/java/scjavalaunch.htm.
http://yourserver/java/scjava.htm.

For Unix systems that have difficulty with the signing mechanism used by the Java client running in a browser, specify a URL to the scapplet.htm file for users to connect to the web server. For example:

http://yourserver/java/scapplet.htm.

Changing the Java Client Heap Size

You should increase the minimum and maximum Java heap size to improve the Java client runtime performance, or if you see OutOfMemoryException messages on the console after you start the client.

To change the Java heap size:

1. Change directories to the Java client installation directory.
2. Change directories to the Java client /RUN.

   Note: The Java client RUN directory is not the same as the ServiceCenter RUN directory. The Java client RUN directory contains one file, **scjava**.

3. Open the file named **scjava** with a Unix editor, such as vi.
4. In the User Configurable Variables section of the **scjava** file, specify a minimum (initial) heap size:

   -ms[size][units]

   Where [**size**] is an integer, and [**units**] is **k** (kilobytes) or **m** (megabytes).
5. In the same section, specify a maximum heap size:

   -mx[size][units]

   For example, to set the initial and maximum heap sizes to 32MB and 48MB respectively, type:

   SCJ_JRE_ARGS="-ms32m -mx48m"

6. Save and close the **scjava** file.

Macintosh Operating Systems

The ServiceCenter installation CD contains installation files for both Mac OS 9.*x* and Mac OS X

Both options are available from the installation start menu. You can install the Java client on a Macintosh in three steps:

1. Install a Macintosh JRE. OSX already has a valid JRE installed. See *Installing the MRJ for OS 9.x*.
2. To install the Java client, see *Installing a Java Client for OS 9.x* on page 46 or *Installing the Java Client: OS X* on page 48.

3 Update the scj.ini file with your host name and server port number. See *Editing the scj.ini file* on page 47.

Installing the MRJ for OS 9.*x*

Peregrine Systems recommends that you use a version 2.2.5 or a later release for best performance. (You can also improve performance by installing additional memory.) You can install a valid MRJ using the installer on the ServiceCenter Java client for Macintosh CD-ROM. The Java client installation program creates a ServiceCenter Java client folder and an MRJ Install folder on your hard drive.

If you have a valid MRJ, continue to the next section, *Installing a Java Client for OS 9.x* on page 46. For more information about platform hardware requirements, see *Peregrine's CenterPoint Web Site* on page 9.

1 Insert the Java client installation CD into your CD-ROM drive. Figure 2-21 shows the initial installation dialog box.

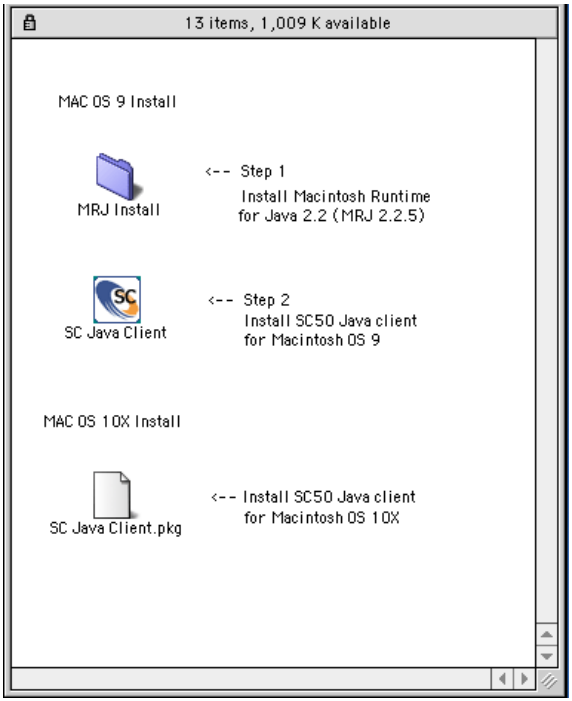


Figure 2-21: Macintosh Installation window

2 Double-click **MRJ Install**. Figure 2-22 shows the MRJ Install window.

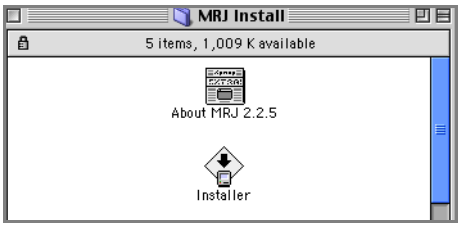


Figure 2-22: MRJ Install dialog window

3 Double-click **Installer**. The legal agreement appears.

4 Click **Agree** in the legal agreement dialog box to continue.

Figure 2-23 shows the dialog box where you can specify a location for the installation. Click **Install** to accept the default location, or click **Switch Disk** to select another location, and then click **Install**. The installation begins.

Figure 2-23: MRJ Installation dialog box

5 When the message appears that the MRJ installation is successful, click **Quit**.

The initial installation dialog box, Figure 2-21 on page 44, reappears.

6 Proceed to step 2 on page 46, *Installing a Java Client for OS 9.x*. It is not necessary to restart your workstation after installing the MRJ.

Installing a Java Client for OS 9.*x*

1 Insert the Java client installation CD into your CD-ROM drive. Figure 2-24 shows the initial installation dialog box.

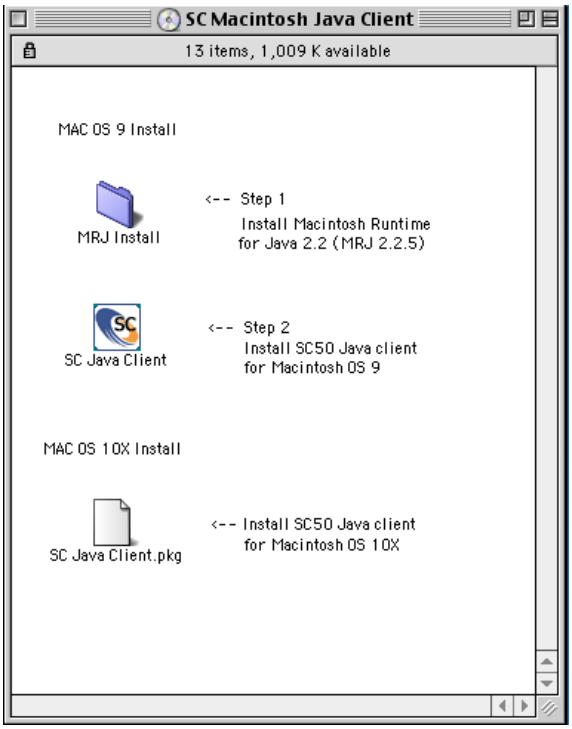


Figure 2-24: Macintosh Installation window

2 Double-click **SC Java Client**. The Java client release notes appear.

3 Click **Continue** after you read the release notes.

4 Figure 2-25 shows the SC Java Client dialog box where you can select the installation location. To install the Java client at the root level of your startup drive, click **Install**. To install in a different location, choose **Select Folder** from the drop-down list, then click **Install**. The installation begins.

Figure 2-25: SC Java Client installation dialog box

5 When the message appears that the Java client installation is successful, click **Quit**.

Editing the scj.ini file

1 Open the ServiceCenter Java client folder shown in Figure 2-26.

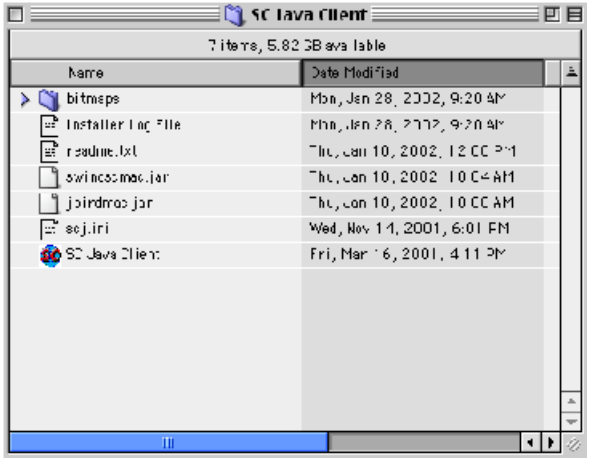


Figure 2-26: ServiceCenter Java client folder

2 Double-click the scj.ini file to open it with the Apple SimpleText editor.

3 Edit line 1. Delete **hostname** and replace it with the host name or IP address of the target ServiceCenter server.

4 Edit line 2. Delete the default port number for a connection to a server, **service:12670**. Replace it with the appropriate port number, for example, **service:12680**.

5 From the File menu, click **Save** to save your changes.

6 From the File menu, click **Quit** to exit SimpleText.

7 Copy the **scj.ini** file to your **System** > **Preferences** folder. If you omit this step, the Java client will not launch in standalone mode.

Installing the Java Client: OS X

Follow the steps in this section to install the Java client on a Macintosh OS X platform. Macintosh OS X has a resident MRJ.

To install an OS X Java client:

1 Insert the Java client installation CD into your CD-ROM drive. Figure 2-27 shows the initial installation screen. Double-click **SC Java client.pkg**.

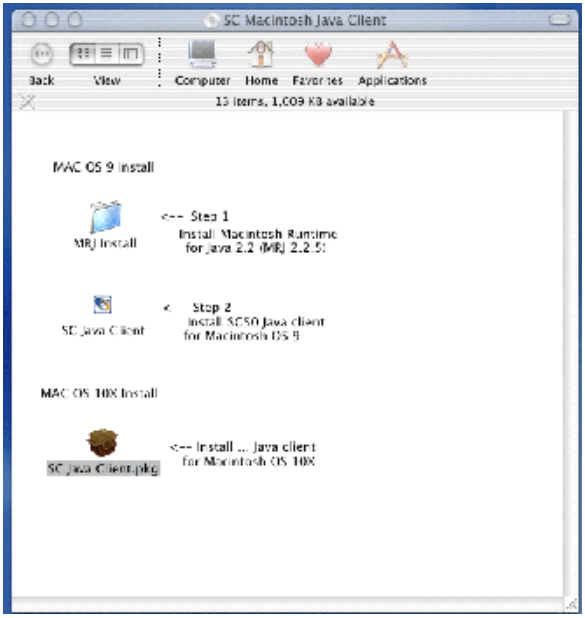


Figure 2-27: Mac OS X installation menu

2 Figure 2-28 shows the Select a Destination window. Select the volume and folder where you want the software to reside. If necessary, click **Choose...** to change folders. Click **Continue** to begin installation.

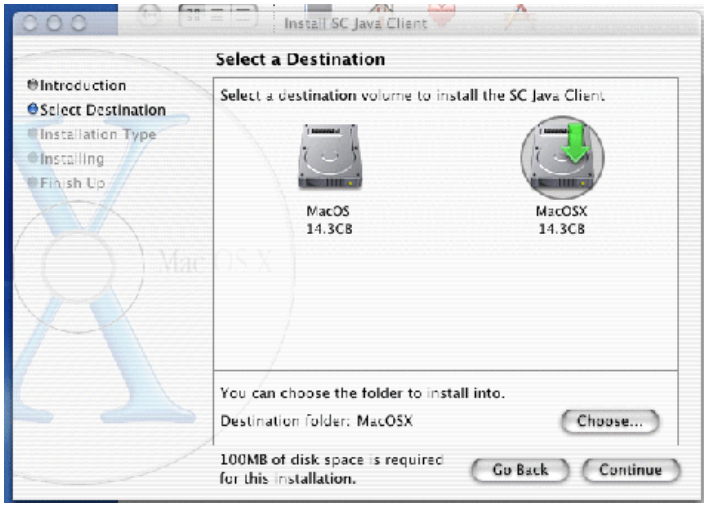


Figure 2-28: Select a Destination

3 Figure 2-29 shows the Easy Install window. Click **Install**.

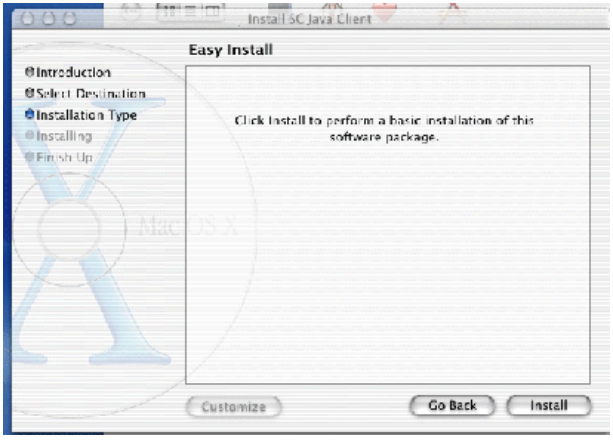


Figure 2-29: Easy Install window

4 The installation wizard notifies you when the installation is complete. Click **Close**.

The client installs into a directory named SC Java Client. Open the sc.ini file from this directory with a text editor. Follow the instructions in *Editing the scj.ini file* on page 47.

Running a Standalone Client

To start the standalone client, double-click the SC Java client icon in the SCJavaClient folder.

Running a Browser-Based Client

To run the Java client in a browser on a Macintosh system, you must connect to a web server using a URL that points to the web server. The web server has a resident scjavalaunch.htm file that subsequently opens the scjavamac.htm file. The scjavamac.htm file appears in the client browser.

For example, you must start a browser session on a Macintosh system that points to http://*yourserver*/path_to_scjavalaunch.htm. on a remote server that is running ServiceCenter server. The scjavalaunch.htm file opens the scjavamac.htm file, which appears in the client browser.

If you need modifications, such as configuring the server hub, you must make them in the scjavamac.htm file.

OS/2 Operating Systems

An OS/2 operating system can host a ServiceCenter Java client, but not a ServiceCenter server.

To install the Java client:

1. Insert the CD into a local CD drive.
2. Browse the CD-ROM and run **../OS2/setup.exe**.
3. During the installation process, specify the directory where the Java client will be installed.

Running a Standalone Client

- Open the OS/2 window console and type the following on a single line substituting your drive letter if necessary.

Note: JRE version 1.1.8 is required. Substitute the installation directory for <jre directory>.

The <space> command indicates where you should press the Space Bar. Do not type <space>.

```
C:\<jre directory>\bin\java<space>-classpath<space>
C:\<jre directory>\lib\classes.zip;C:\<javaclient
directory>\java\jbird.jar;C:\<javaclient directory>\java\swingsc.jar<space>
com.peregrine.sc.client.ClientApplication<space>
-host:<host><space>-service:<port><space>
-imagepath:C:\<javaclient directory>\java\bitmaps<enter>
```

Installing the Java Runtime Environment

You can use Sun's Get Java web site to obtain newer JRE versions. JRE version 1.3 has a performance advantage over earlier JRE versions. When you install this version on the server and select it as the default JVM, a ServiceCenter Java client can connect to the ServiceCenter server using the standard **scjavalaunch.htm** file.

To install the JRE:

1 Connect to *http://java.sun.com/getjava*.

2 Follow the instructions to download the Get Java plug-in. The setup files download and the installation starts automatically. This plug-in becomes a default browser JRE if you use Internet Explorer. Netscape versions 6 and later releases also use the Java plug-in as the default JRE. This means the HTML files that launch applets do not need modification to explicitly use the Java plug-in.

If the ServiceCenter Java client has JRE version 1.3, and it starts with **scjava13plugin.htm**, the Java client uses the local JRE version 1.3. If you do not have a local JRE 1.3, a prompt appears to download and install the JRE version 1.3 from Sun.

To use JRE version 1.3:

1. Install JRE version 1.3 on the server.
2. Request users to connect to ServiceCenter through the scjavaplugin13.htm file.

   If users do not have JRE version 1.3 installed and they connect to the standard scjavalaunch.htm file, the ServiceCenter Java client starts using the browser's default JVM.

Connection Speed

You can test the network connection between your ServiceCenter server and the Java client. Press Ctrl+Shift+S to launch a test that sends small packets of data from the client to the server. This test lasts approximately 15 seconds. The results appear in an Active Notes window to report the number of server transmissions per second and the approximate bandwidth available for the Java client. The output information is also available in the Java console.

The Java Console

The Java Console has configuration settings that you can customize from the workstation Control Panel. There is also an interactive Java Console window that runs from the browser.

Java Console settings on the Control Panel

To view the Java Console:

1. From the **Start** menu, click **Settings- >Control Panel > Java Plug-in**.
2. Click the **Browser** tab. The Java plug-in is the default JRE for Microsoft Internet Explorer. If you change this tab, you must restart your workstation. Figure 2-30 on page 53 shows the Java Console with the Java plug-in selected for Microsoft Internet Explorer.

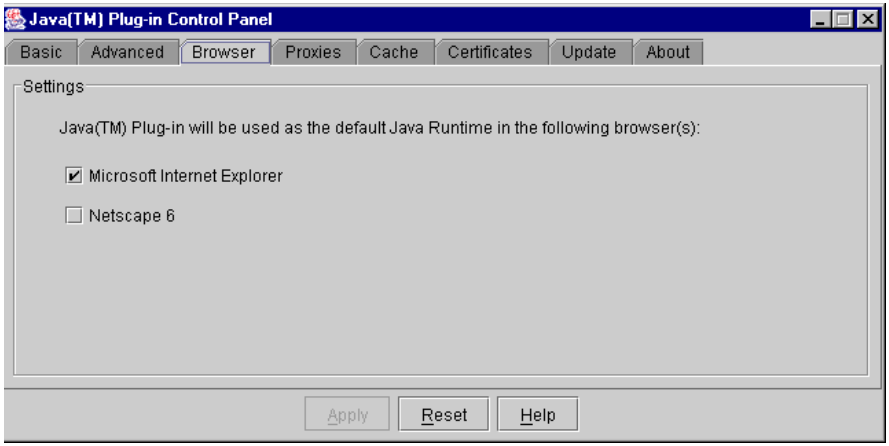
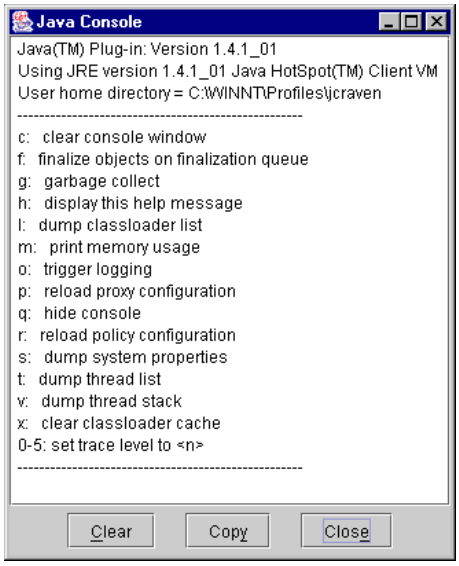


Figure 2-30: Java Console

To view the Java Console window:

1 Start your browser.

2 From the **Tools** menu, select **Sun Java Console**. A coffee cup icon appears in the system tray and the Java Console window appears with a list of commands that you can issue during the browser session.



CHAPTER 3 Configuring the Client

The Java client relies on parameters and user preferences to run successfully in your environment. All parameter names are set in the **scjava.htm**, **scapplet.htm**, or **scjavamac.htm** file, depending on your system. you can pass standalone application parameters on the command line.

Read this chapter for information about:

Java Client .htm Files

To run the Java client from a browser, the browser loads an HTML file that references the appropriate Applet code and provides specific parameters necessary for execution, including those outlined in the previous section. For more information, see *Running a Browser-Based Java Client* on page 34.

The following example shows the applet code in the **scjava.htm** file.

```
<applet
codebase=http://www.company.com
code=com.peregrine.scinstaller.ClientInstaller
archive=sc.jar
width="100%"
height="100%"
vspace=0
hspace=0
align=middle>
<param name="cabbase" value="sc.cab">
<param name="Host" value="127.0.0.1">
<param name="Service" value="12670">
<param name="Language" value="en">
<param name="codeset" value="English">
<param name="Timeramount" value="15">
<param name="ImagePath" value="http://www.company.com/bitmaps/">
</applet>
```

You should understand the different parameters in this file so you can make necessary changes. The more significant parts of this file are described below.

```
codebase=http://www.company.com
```

Specifies the URL where the Java client files reside. This example points to a web server location. If you install the files to a local drive, this parameter should contain the directory path. For example:

```
codebase="file:///C:\Programs\...\...\Java"
<param name="Host" value="127.0.0.1">
```

Specifies the host IP address of the ServiceCenter server. This address must be accessible by user machines. If the host is inside a company firewall, only users within the firewall can run the client remotely, unless you are running the Server Hub component. For more information, see *Server Hub* on page 73. Type your server IP address. The default value of 127.0.0.1 translates to localhost.

```
<param name="Service" value="12670">
```

The default port used by the ServiceCenter server is 12670.

```
<param name="ImagePath" value="http://www.company.com/bitmaps/">
```

Specifies the location where the client image files reside. This example points to a web server location. If you install the files to a local drive, this parameter should contain the directory path. For example:

```
<param name="ImagePath" value="file:///C:\Programs\...\...\bitmaps">
```

During installation, the client bitmaps are automatically downloaded and installed on the remote client workstation. This parameter specifies a bitmap directory to be used if a bitmap file cannot be found on the Java client workstation.

Setting Preferences in .htm Files

There are optional parameters that can you can specify for the Java client in the .htm file you use to launch the client. The format for each parameter is:

```
<param name=”parameter” value=”value”>
```

Parameter Name	Function
envdump	Generates a diagnostic dump of the Java client environment. When you specify this parameter, the client creates the dump and exits immediately.
codeset	Specifies the language to be used. You must specify a valid two-character ISO identifier, such as fr for French. For a complete list, see *Language Indicators* on page 71.

Parameter Name	Function
language	Specifies the language environment, such as English or Japanese, when you view the Java client. You must specify a full ISO language identifier, such as French or English. For a complete list, see *Language Indicators* on page 71.
scjpath	Specifies the path to the scj.ini file. The scj.ini file is an optional file that you can create and save in the directory specified by this parameter. For more information, see *Setting Preferences in Scj.ini* on page 59.
scjpref.ini	If there is an scjpath file, there is a generated scjpref.ini file that stores user preferences.
DictionaryDir	Specifies the directory that contains the dictionaries used by spell checkers. If you do not want to use the default dictionaries provided by the client, set the DictionaryDir parameter to point to the dictionary directory that you want to use. Specify a URL or an absolute path.
DownloadDictionaryDir	Specifies a directory where you can download the dictionaries to your workstation. When the dictionary download is complete, the Java client writes the directory information to the scj.ini file. Specify a URL or an absolute path.

Dictionarydir and Downloaddictionarydir Parameters

If you have a default dictionary on your workstation, the Java client uses that dictionary for spell checker operations. If the client cannot find a local dictionary, it searches for a dictionary using these methods:

- The client looks at the location specified by the dictionarydir parameter. If the client finds a dictionary in the specified location, it uses that dictionary for spelling checker operations.
- If the client cannot find a dictionary in the location specified by the dictionarydir parameter, it looks at the downloaddictionarydir parameter for a source to download dictionary files. When the download operation is complete, the system writes the local dictionary location to the scj.ini file to prevent the client from downloading dictionary files again.

Setting Preferences in Scj.ini

You can specify parameters in the command line, an .htm file, or in the scj.ini file. The scj.ini file is an optional file that you can create and save in the directory specified by the scjpath parameter. If the applet locates the scj.ini file, it uses the parameters in that file instead of those in the .htm file when it launches the client. You can review the description for the scjpath parameter in *Setting Preferences in .htm Files* on page 57.

If you create an scj.ini file, store it in the directory specified by the User Preferences. The default installation directory is C:\Program Files\Peregrine\ServiceCenter\Java. You can find the User Preferences directory for your system by clicking **Help > About ServiceCenter**. Figure 3-1 shows the scj.ini file path.

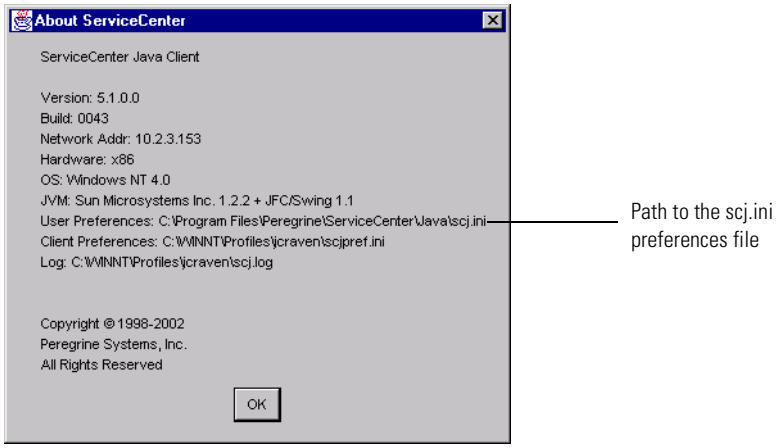


Figure 3-1: User Preferences directory

The Java client uses the parameters in this file only if you did not specify them on the command line.

Format of the scj.ini file

The format for each parameter in the scj.ini file is:

name:value

A line that starts with a pound (#) sign is a comment.

For example:

```
# This is a comment
host:www.mycompany.com
service:12345
imagepath:http://www.mycompany.com/images
```

Other scj.ini parameters

You can redirect the Java client by adding the host and service variables to the scj.ini file. The scj.ini file overrides the settings in the .htm launch file. If the scjpath parameter is in the .htm launch file, when a Java client connects, it uses the host and service variables specified in the scj.ini.

Parameter	Definition
host	Specifies the host name or IP address of the host.
service	Specifies the ServiceCenter service name or number for the Java client connection. For example, the default Service ID is 12670.

Setting Preferences in Scjpref.ini

This file contains internal preference settings used by the Java client. The Java client reads and updates this file at run time as necessary. The Java client uses this file to store runtime information such as window placement, font and color preferences, table column sizes, and so forth. The directory where this file resides depends on your platform and installation. You can find the User Preferences directory for your system by clicking **Help** > **About ServiceCenter**.

Figure 3-2 shows the scjpref.ini file path.

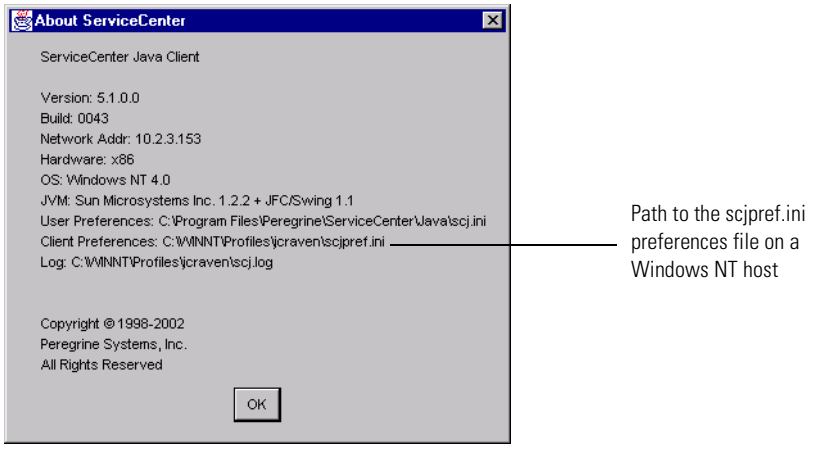


Figure 3-2: Client Preferences directory

Setting Preferences in Sc.ini

The ServiceCenter server can control certain Java client preferences with values that appear in the server sc.ini file. If you specify a value the server does not recognize, the default value is 1. If you insert a parameter without a value in the sc.ini file, the server uses the value specified in the scjpref.ini file, if the parameter and its value exist there.

If you make a change to a ServiceCenter .ini file, you must restart the server to activate the change.

The syntax for these parameters is: parameter:variable

Parameter	Variable
clientprinting	Specify 0 or *n*, where *n* is a positive number greater than zero.
explorer	Specify 0 to disable the ServiceCenter Explorer feature, or 1 to activate the feature. For more information, see *Explorer Parameter* on page 63.

Parameter	Variable
explorerdefault	Specify 0 for the Menu mode to appear at startup. Specify 1 for SC Explorer mode. In menu mode, you can select SC Explorer from the View menu. At startup for a new installation, the default view is SC Explorer mode. For more information, see *Explorerdefault Parameter* on page 63.
explorerhome	Specify the name of the start menu if you create a custom menu. The default value is ***string1***. If you specify an invalid value, ServiceCenter ignores it and uses the default value. For more information, see *Example: Explorerhome* on page 65. For example, if you create a new menu named java, specify: explorerhome:java
menuforms	Specify 0 to disable the Menu Forms feature, or 1 to activate the feature. If the **explorer** and **menuforms** values are the same, such as 0 and 0, or 1 and 1, ServiceCenter ignores both values and uses the last view setting. Peregrine Systems recommends that you specify a value of 1 for one parameter, and 0 for the other parameter, depending on which you prefer to use.
nohelponfield	Specify 0 to disable the help feature, or 1 to activate the feature.
sctimeramount	Specify *n*, where *n* is a positive number greater than zero. For example, if you want the server to check each client every 20 seconds, set the parameter to 20 in the **sc.ini** file: sctimeramount:20
viewactivenotes	Specify 0 to disable the active notes feature, or 1 to activate the feature.
viewattachments	Specify 0 to disable the attachments, or 1 to view the attachments. The attachments exist with the record, but users cannot view or attach files when this parameter is disabled.
viewpromptforsave	Specify 0 to disable the prompt to save a record when you make changes, or 1 to activate.
viewrecordlist	Specify 0 to disable record lists, or 1 to view the record lists.
viewtoolbar	Specify 0 to hide the common toolbar, or 1 to display the toolbar.
viewtraycaptions	Specify 0 to disable the tray captions, or 1 to view the captions.

Explorer Parameter

Settings in the sc.ini and scjpref.ini files can affect the Java client view mode at startup.

Case 1

There is no explorer parameter specified in the sc.ini file.

If the scjpref.ini file exists, you will see the last selected view at startup because the Java client stores the last setting selected in this file. You can manually change the view mode by selecting or deselecting SC Explorer from the View menu.

If the scjpref.ini file does not exist, you will see SC Explorer view by default. You can manually change the view mode by selecting or deselecting **SC Explorer** from **View** menu.

Case 2

The sc.ini file has this parameter: explorer:0.

Whether the scjpref.ini does (or does not) exist, you will see only the Menu Forms view. The View menu will not display an SC Explorer option to select.

Case 3

The sc.ini file has this parameter: explorer:1.

Whether the scjpref.ini does (or does not) exist, you will see only the Explorer view. The View menu will not display an SC Explorer option to deselect.

Explorerdefault Parameter

Settings in the sc.ini and scjpref.ini files can affect the Java client view mode at startup.

Case 1

There is no explorerdefault parameter specified in the sc.ini file.

If the scjpref.ini file exists, you will see the last selected view at startup because the Java client stores the last setting selected in this file. You can manually change the view mode by selecting or deselecting **SC Explorer** from the **View** menu.

If the scjpref.ini file does not exist, you will see SC Explorer view by default at startup. You can manually change the view mode by selecting or deselecting **SC Explorer** from the **View** menu.

Case 2

The sc.ini file has this parameter: explorerdefault:0.

Whether the scjpref.ini file does (or does not) exist, you will see the Menu Forms view at startup because the setting in the sc.ini file overrides all other settings. You can manually change the view mode by selecting or deselecting **SC Explorer** from the **View** menu.

Case 3

The sc.ini file has this parameter: explorerdefault:1.

Whether the scjpref.ini file does (or does not) exist, you will see the SC Explorer view at startup because the setting in the sc.ini file overrides all other settings. You can manually change the view mode by selecting or deselecting **SC Explorer** from the **View** menu.

Example: Clientprinting

If you specify clientprinting:0, all clients logged on to that server use the server for printing. The Print on Client option on the Java client tool bar is unavailable. The Print on Server option is checked.

If you specify clientprinting:10, all clients logged on to that server use client printing and the page limit of the client report is 10 pages. The Print on Client option on the Java client tool bar is checked. The Print on Server option is unavailable.

Setting any of these parameters through the server disables the corresponding menu option on all clients, not just the Java client. Clients cannot override the server setting.

Note: The Java client does not support limiting the client report to *n* pages if the client is currently printing.

Example: Explorerhome

System administrators can customize the ServiceCenter Explorer menu by creating a new start menu, modifying a user's operator record to point to the new start menu, and adding the explorerhome parameter to the sc.ini file. You can use the Java client itself to perform these steps and restart the Java client to see the results. The basic steps are to:

- Create a new root level menu.
- Create a Menu record
- Add the explorerhome parameter to the sc.ini file.

To create a new root level menu:

1. From the ServiceCenter main menu, click the **Utilities** tab.
2. Click the **Administration** button. The administration panel appears.
3. Click the **Operators** button in the **Security** group.

4 Type a Login Name (such as falcon) for the operator record that you want to modify. Press **Enter**. Figure 3-3 shows the Operator Record.

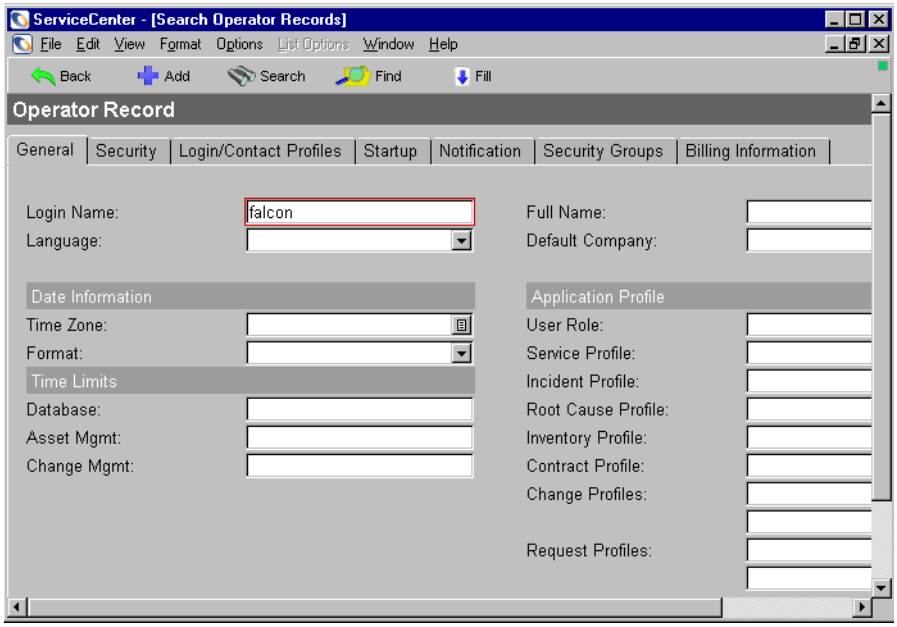


Figure 3-3: Search Operator Records

5 Click the **Startup** tab. Specify a start menu in the Initial Application box.

6 Type a new **Parameter Name** in the next available field. For example, **java**.

7 Type a value in the **Parameter Value** field. For example, JAVA HOME. Figure 3-4 on page 67 shows the Initial Application area with a new Parameter Name and Parameter Value added.

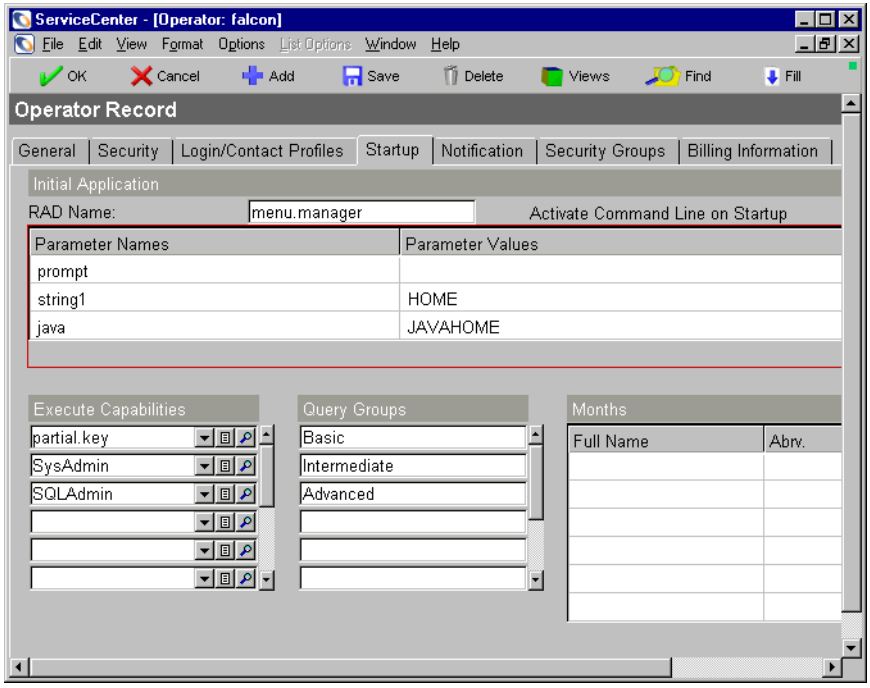


Figure 3-4: Initial Application information

Warning: The Parameter names **name** and **string 1** are available in out of the box systems. Do not modify these parameters. You can damage your system when editing the values for the **name** or **string1** parameters.

8 Click Save.

To create a Menu record:

1 From the ServiceCenter main menu, click the **Utilities** tab.

2 Click the **Tools** button.

3 Click the **Menus** button.

4 Type a menu name in the **Menu Name** field to open a menu that you can copy. Figure 3-5 shows an example using the HOME menu. Press **Enter**.

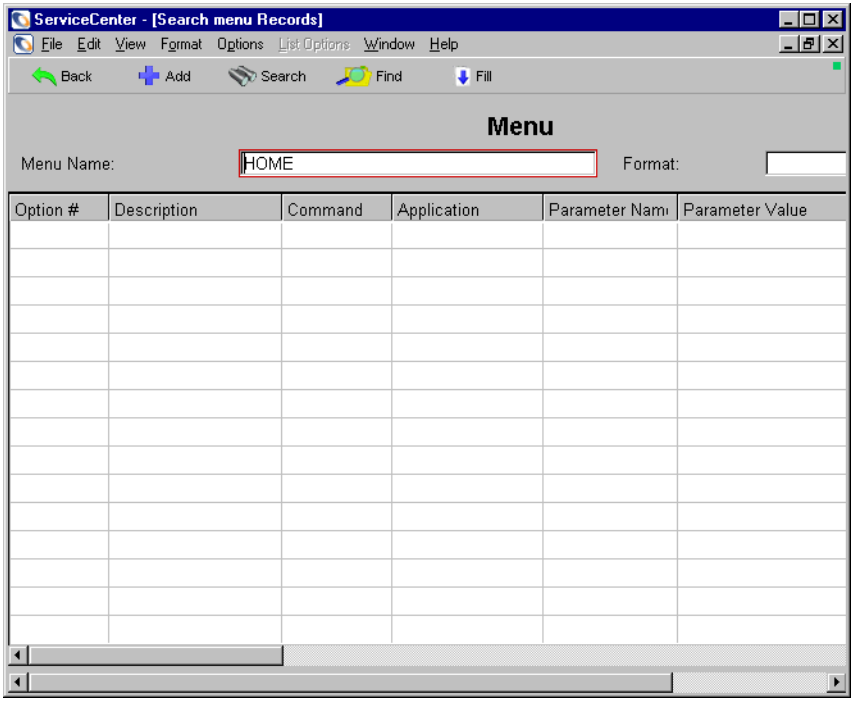


Figure 3-5: Initial Application information

5 Create a new menu based on the current menu by renaming it. Type a new name in the **Menu Name** field, such as JAVA HOME, and click Add to save the new menu. Remember to use the same menu name you specified in step 7 on page 66. Figure 3-6 on page 69 shows the renamed menu.

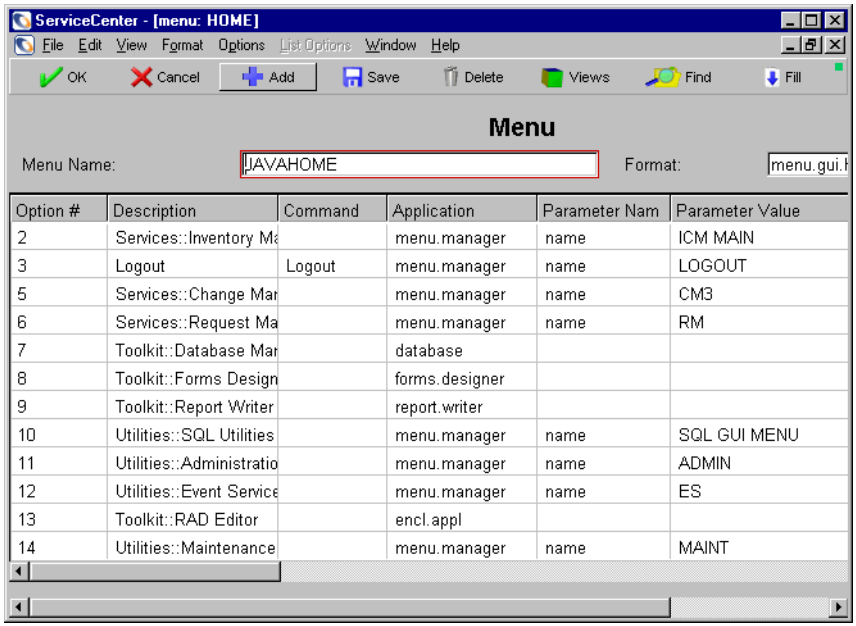


Figure 3-6: Initial Application information

6 You can edit any field in the **Description** column to change the text that displays in the SCExplorer menu.

7 You can edit any of the fields in the **Command** column to change the behavior of the associated command.

8 Click **Save**.

To add the explorerhome parameter to the sc.ini file:

1 From the Windows **Start** menu, select **Programs** > **Peregrine ServiceCenter** > **Init File**. Your default text editor opens \\Peregrine\ServiceCenter\RUN\sc.ini.

2 Add (or modify) the explorerhome parameter:

```
explorerhome:java
```

The value for the explorerhome parameter is whatever you specified in step 7 on page 66.

3 Save the changes and close the file.

Notes

To modify any branch of the Explorer menu, repeat this process for the branch menu.

- To create a new root level menu, see *page 65*.
- To create a Menu record, see *page 67*.
- To add the **explorerhome** parameter to the **sc.ini** file, see *page 69*.

For example, edit the SLA menu by entering **SLA** in the menu search, and then modify the values for that menu. Peregrine Systems recommends that you rename and add any new menus before you edit them.

Setting Preferences at the Command Line

When you run the standalone Java client, the application invokes the JRE installed on your system to execute the ServiceCenter client classes. Most of the applet parameters can be passed on the command line to the standalone application.

You can start the standalone Java client with this command:

```
c:\path_to_jre -classpath c:\path_to_library\classes.zip;jbird.jar;swingall.jar
com.peregrine.sc.client.Client Application -host:127.0.0.1 -service:12670
-imagePath:c:\path_to\BITMAPS
```

where the classpath points to the class library files for the JRE runtime library.

This command starts the JRE and loads the main ServiceCenter class (**com.peregrine.sc.client.ClientApplication**) for execution. The last three parameters are required.

Parameter	Definition
-host:*host_ip*	Specifies the host IP address of the ServiceCenter server. This address must be accessible by the workstation. If the host is inside a company firewall, only users within the firewall can run the client.
-service:*service*	Specifies the service used by the ServiceCenter server.
-imagepath:*bitmap_path*	Specifies the location of the client image files. Image files are normally copied and cached by the Java Installer. For more information, see *Running a Browser-Based Java Client* on page 34.

Record List Auto Refresh

A record list appears in a split interface at the top of a window. The Java client can update record lists with new records on a set schedule. The Java client queries the server and displays any changes in the record list to the user.

To set the auto refresh rate for a record list:

1. Open a record list form from the Forms Designer utility, for example probsummary.qbe.g. For more information about the Forms Designer utility, see System Tailoring, Volume 1.
2. Type the name of the form in the **Form** field. For example:

   probsummary.qbe.g.

3. Select the record to update in the Forms Designer utility and press **F9** or click the **Design** button. The Forms Design mode appears.
4. Click inside the table to display the Properties box.
5. In the Properties box, select **RefreshRate**.

   Type a refresh rate in seconds. A refresh rate of 0 disables this feature. If you specify a very high refresh rate (less than 15) may affect your ability to interact with the Java client.

Language Indicators

The Java client version 5.1 has localized versions in French, German, and Italian; however, it can accept data in Chinese (simplified and traditional), Japanese, Korean, Polish, Thai, Turkish, and ISO Latin1. The following table lists the ISO two character indicators for the languages that the Java client supports. For more information, see the *ServiceCenter Technical Reference*.

Language	Identifier
Chinese	zh
French	fr
German	de
Italian	it
Japanese	es
Korean	ko

Language	Identifier
Polish	pl
Thai	th
Turkish	tr

Example: Standalone Turkish Language Support

To use the Java client as a standalone application with Turkish, modify the shortcut to start the client.

1. Locate the Java client shortcut icon on the Windows desktop.
2. Right click and select **Properties** from the Shortcut menu. The Properties dialog box appears.
3. Click the **Shortcut** tab.
4. In the Target text box, add the following to the end of the command:

   -codeset:tr -language:Turkish

5. Click **OK**.

4 Server Hub

CHAPTER

The server hub is a Java servlet that adds an additional security layer to the Java client by acting as a proxy to the ServiceCenter server. Instead of connecting directly to the ServiceCenter server, the Java client identifies itself to the server hub, usually through an HTTP connection. Direct TCP/IP connections to both the client and the ServiceCenter server are through the server hub, which relays information between the client and server.

The client does not need to know the location of the ServiceCenter server; therefore, the ServiceCenter server can run on a different host than the server hub. The server hub can also be used in conjunction with a firewall, and can be used with multiple ServiceCenter servers. Whether the Java client connects directly to a ServiceCenter server or connects indirectly through the server hub depends on client-side parameters, the connection details are transparent.

Read this chapter for more information about:

Server Hub Requirements

There are four components that you work with to install and run a server hub:

- A web server, which is optional if you run a standalone servlet engine that is capable of initializing servlets without an HTTP server connection
- Servlet engine
- Java client
- ServiceCenter server

The installation installs the server hub file as part of the standard Java client installation. It consists of a single Java JAR file: **serverhb.jar**. The default path to this file is \\Peregrine\ServiceCenter\ java**serverhb.jar**. To configure the Java client/server hub connection, there are three tasks:

- Configure the web server for servlet zones.
- Configure the servlet engine to include the **serverhb.jar** file.
- Configure the ServiceCenter Java client to connect to the server hub.

The server hub servlet package that is passed to the servlet engine is **com.peregrine.hub.HubServlet**. This package name is case sensitive.

Using Java Servlets

The server hub is a Java servlet. Java servlets are server side Java programs that can generate dynamic content. They support a request/response model that is commonly used in servers. They are more efficient than CGI programs, which start a new process each time they are invoked. From the client side, you can access a Java servlet in the same way as a standard CGI script. Java servlets are platform-independent as well as faster and more secure than CGI scripts.

Servlet Support Requirements

The server hub requires an HTTP server that supports Java servlets, or a standalone servlet engine that supports servlet zones without using an HTTP server. Java servlet extensions to most major web servers are available for a variety of platforms. Several standalone servers offer Java servlet support as well. You can find more information about Java servlets, servers, and platforms, at *http://java.sun.com*.

Server Hub Parameters

The following table lists server hub connection parameters that apply to the server hub servlet. You can set these parameters when you configure the servlet engine.

Parameter	Description
servers	Required. Specifies a semicolon-delimited list of one or more ServiceCenter servers that clients may connect to through the server hub. Specify each server using the syntax host:port, with an optional alias or nickname used by the clients to identify the server. By using an alias for each server, the network location of the server is transparent to the client. To set this parameter using both types of server specifications: servers=server1:12670;mainserver(server2:12670) The client must identify the first server as server1:12670, the second as mainserver.
default	Optional. Specifies the name (from the servers lists) of the default ServiceCenter server if the client does not specify a server. For example: default=mainserver
threads	Optional. Specifies the maximum number of threads allocated to the server hub to handle client connections. This value specifies the maximum number of simultaneous users supported by the server hub. If you omit this parameter, the default value is 500. For example: threads=500
connectport	Required. Specifies the port number on the server running the server hub that can be used by the client to connect directly to the server hub, bypassing the standard callback mechanism. This parameter is optional. The direct connection feature is disabled if you do not specify a port. For more information, see *HTML File Parameters* on page 76. For example: connectport=4331

HTML File Parameters

The following table lists the connection parameters that you can insert in the HTML file that you use to launch the Java client (**scjava.htm**, **scjavamac.htm** or **scjavaplugin.htm**). These parameters also apply to the standalone Java client. Set parameters as necessary, depending on your configuration.

The HTML parameter syntax is:

```
<param name="enter a parameter" value="give an appropriate value">
```

Parameter	Description
HubURL	Required. Specifies the URL of the server hub. For example: http://your_web_server:8080/hub/servlet/HubServlet Setting the HubURL parameter overrides the Java client **Host** and **Service** parameters. This parameter can also specify the server hub server ID and port for a direct **connection type**. For example: servername:12345 If the HubURL parameter specifies the server hub URL, the server hub will be queried for the host name and port for direct connections.
HubAdapter	Optional. Specifies the type of server connection: SCExpress or SCExpressSL. SCExpressSL does not return some status codes to increase speed. If your network has a slow response time, specify SCExpressSL. The default value is SCExpress.
Server	Required. Specifies the server name or server hub alias of the ServiceCenter server that you specified in the server hub servers parameter. For example, mainserver. This parameter is optional only if you set the server hub default parameter.
ClientPort	Optional. Specifies the port number on the client machine for the client to use when it connects to the server hub. The port must be available to the client.
ConnectionType	Optional. Specifies the type of connection between the server hub and the client. You can specify callback or direct. The default value is callback. A direct connection enables the client to initiate the connection to the server hub if the client is running behind a firewall or on a machine that does not have a static IP address. A callback connection specifies that the server hub randomly assigns a connection port to the Java client. The client must have a static IP address for this connection to work.

Installation Scenarios

How you install and configure the server hub depends on how the client accesses the server hub. For example, the server hub allows clients outside a firewall to connect to a ServiceCenter server inside a firewall. If there is no firewall between the client and ServiceCenter server, you do not need the server hub to connect the Java client and ServiceCenter server. If you must install and configure the server hub, the primary consideration is whether the client IP address is static or uses Network Address Translation (NAT).

There are three types of server hub connections that you can install and configure:

- Use a callback connection when the Java client is installed on a workstation with a static IP address. See *Callback Connection*.
- Use a direct connection when you launch the Java client from a workstation with a dynamic IP address using NAT. See *Direct Connection* on page 86.
- Use a direct connection without an HTTP server when the servlet engine can run without an HTTP server. See *Direct Connection Without an HTTP Server* on page 89.

In each type of server hub connection, there are three steps. First, you must configure the servlet engine. Next, you must configure the Java client. Finally, you may need to configure the Java client to connect through one or more firewalls.

Callback Connection

This is a simple connection used when the Java client runs on a workstation with a static IP address. This is the sequence of events:

- The Java client communicates its IP address and port to the server hub using an HTTP connection.
- The server hub opens a TCP/IP connection back to the specified address.
- The server hub opens a TCP connection to the ServiceCenter server.
- The server hub runs transparently as a proxy server.

Figure 4-1 shows the callback connection from the Java client to the server hub.

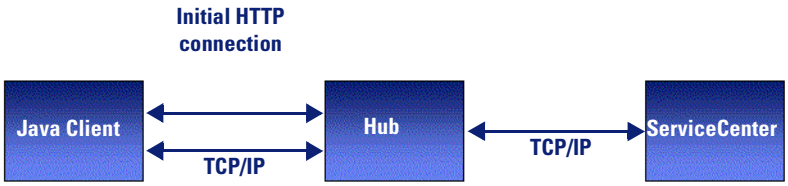


Figure 4-1: Direct connection

Step 1: Configure the Servlet Engine

The first step is to configure the servlet engine to work with the server hub. The first example uses Tomcat. The second example that begins on page 81 uses JRun.

Tomcat example

Apache Tomcat is an open source servlet engine available for most platforms. The Java client requires Tomcat 4.1 or a later release. For more information, see *www.apache.org* for more information about the Apache web server and Tomcat servlet engine. The Tomcat servlet engine does not require a web server to handle servlets if you configure it as a standalone servlet engine.

Install and configure Tomcat by following the product documentation. In the following example, TOMCAT_HOME is the Tomcat installation directory.

To configure a Tomcat servlet engine:

1 To deploy the Peregrine server hub, you must modify the Tomcat directory structure. Under TOMCAT_HOME/webapps, add a new directory named hub for the hub servlet. Under the hub directory, add a sub-directory named WEB-INF and a Web-inf subdirectory named lib. Figure 4-2 shows the additions to the Tomcat directory structure.

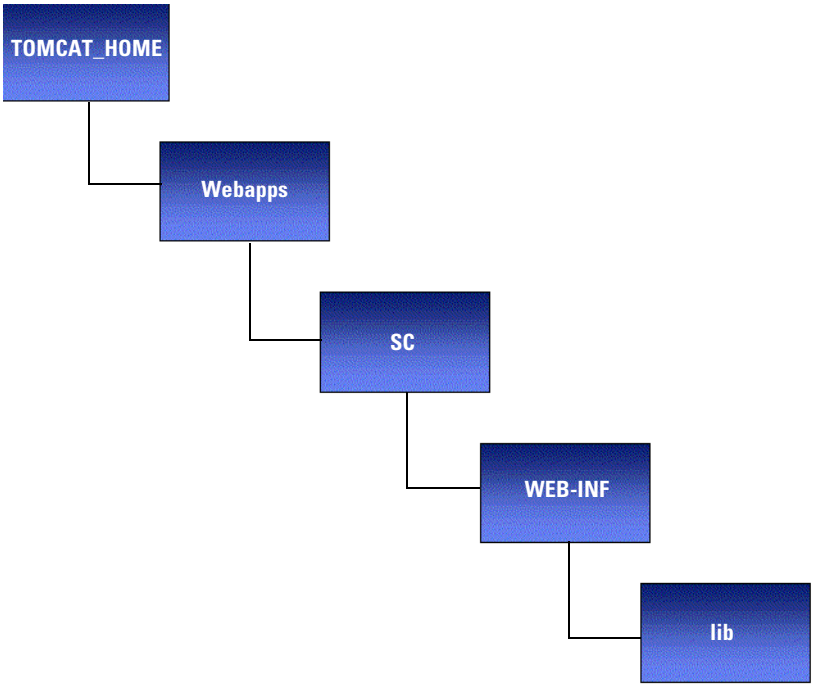


Figure 4-2: Configure the Tomcat directory

2 Use a text editor to create a file named web.xml and store it in the Web-inf directory. The web.xml file specifies the server hub parameters. The following is an example web.xml file with the servers and default parameters set. For more information, see *Server Hub Parameters* on page 75.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
      <servlet-name>Hub</servlet-name>
      <servlet-class>com.peregrine.hub.HubServlet</servlet-class>
          <init-param>
              <param-name>servers</param-name>
              <param-value>mainserver(your_servicecenter_server:12670)
              </param-value>
          </init-param>
          <init-param>
              <param-name>default</param-name>
              <param-value>mainserver</param-value>
          </init-param>
      <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
      <servlet-name>Hub</servlet-name>
      <url-pattern>/*</url-pattern>
</web-app>
```

3 Copy the \\...\ java\serverhb.jar file from the installation directory on your workstation to the lib directory.

4 Restart Tomcat.

5 Assuming you followed the directory naming conventions in the example, test your configuration by connecting to:

```
http://your_web_server:portnumber/hub/servlet/Hub?cmd=1
```

If the server hub is working correctly, you should see a blank page or a dialog box that says the page contains no data. If not, review the steps, all configuration settings, and confirm that you set the classpath correctly. The parameter to insert in the HTML file that you use to launch the Java client is:

```
<param name="HubURL"
value="http://your_web_server:portnumber/hub/servlet/HubServlet/">
```

Notes

1. To connect to the server hub, you must specify the HubURL parameter value in the HTML file that launches the Java client. For more information, see *Step 2: Configure the Java Client* on page 83.
2. If Tomcat runs as a standalone application, you can specify the server hub URL without passing a port number. For example, you can use a URL like http://localhost/servlet/hub instead of http://localhost:8080/servlet/hub.

JRun 3.1 example

JRun is a commercial engine that installs two web servers by default. You must configure JRun to be aware of a third party web server such as Apache or IIS. For more information about JRun, see *www.macromedia.com*.

To configure a JRun servlet engine:

1. Install and configure JRun using the JRun documentation.
2. Start the JRun administrator tool.
3. Open the **JRun Default Server** branch shown in Figure 4-3 on page 82.
4. Click **Java Settings**. The Java settings appear in the right pane.
5. Click **Classpath** and add the path to the serverhb.jar file.
6. Click **Update**.
7. After you add the serverhb.jar file to the JRun class path, you must tell JRun how to start the server hub. Click **Web Applications** in the menu tree of the JRun administrative tool. The Deploy Web Applications wizard appears in the right pane. Use the Deploy Applications Wizard to create a web application for the server hub.

   Note: Use the Wizard to define a URL mapping for the server hub. The mapping informs the web server that requests for certain URLs, such as /hub, should be handled by JRun and not the web server.

8 The web application created in step 7 on page 81 will be empty. To define the server hub parameters, expand the Web Applications tree and select the Web application you created in step 7, as shown in Figure 4-3.

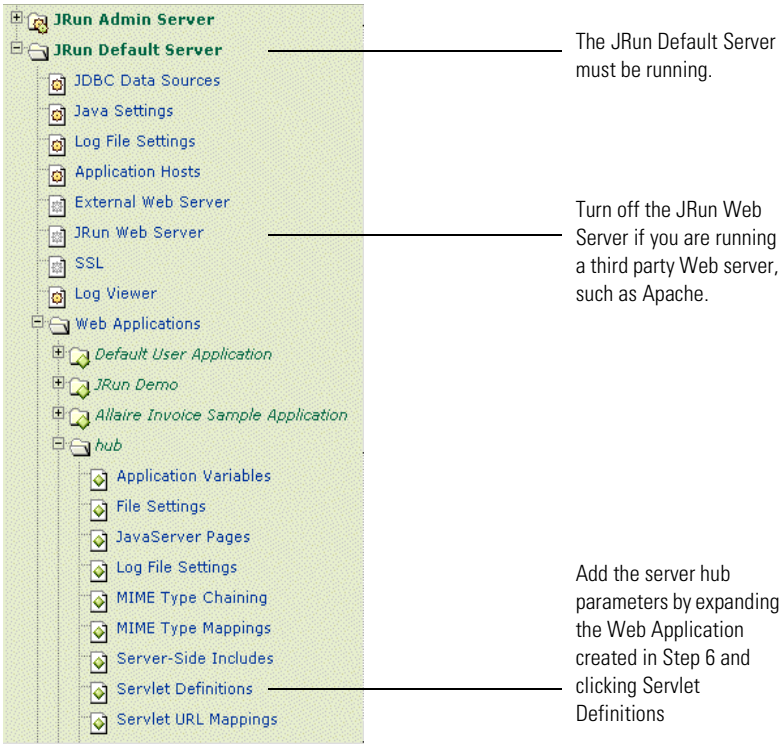


Figure 4-3: JRun configuration tree

9 Click **Servlet Definitions.**

10 Click **Edit** on the right-most panel. The JRun edit window appears.

11 In the JRun edit window, type the following values:

- In the **Name** field, type **hub.** In the **Class Name** field type:

  com.peregrine.hub.HubServlet

- Scroll the JRun edit window to the right to view the **Init Arguments** field. Type the server hub initialization parameters in the **Init Arguments** field. For example:

  servers=mainserver(Your_servicecenter_server:12670)
  threads=500
  default=mainserver

For a complete description of the Init parameters and values, see *Server Hub Parameters* on page 75.

12 Restart your web server.

13 Assuming you followed the directory naming conventions in the example, test your configuration by connecting to:

http://www.yourserver.com/hub/servlet/HubServlet?cmd=1

If the server hub is working correctly, you should see a blank page or a dialog box that says the page contains no data. If not, review the steps, all configuration settings, and confirm that you set the class path correctly.

Step 2: Configure the Java Client

There are two methods that you can use to launch the Java client. If you choose Method 2, configure the HTML file that you use to launch the Java client with the connection port. The HTML files are scjava.htm, scjavamac.htm or scjavaplugin.htm.

Method 1 — The Java client randomly selects a connection port to the server hub. Choose this method when there is no firewall between the client and server.

Method 2 — Configure the HTML file that launches the Java client to specify the connection port. Choose this method when there is a firewall between the client and server. The connection port defined in the HTML file must be open for incoming traffic. For more information about the correct port, see your local system or web administrator.

The Java client must be aware of the server hub to make the connection to the hub. The following settings are made in the HTML file that you use to launch the Java client, or are passed on the command line for the standalone application.

Method 1 parameters

Use the parameters in the following table when the Java client is not behind a firewall and can select a random connection port.

Parameter	Definition
HubURL	Specifies the URL of the server hub. For example: <paramname="HubURL" value= "http://your_web_server:port_num/hub/servlet/HubServlet"> Setting the HubURL parameter overrides the Java client host and service parameters.
HubAdapter	Specifies the type of server connection to be made. Specify SCExpress.
Server	Specifies the server name or server hub alias of the ServiceCenter server to connect to that you defined in the server hub servers parameter (for example, mainserver). This parameter is optional if you specify the server hub default parameter.

HTML file example

Replace the Host and Service parameters with the following parameters in the HTML file that you use to launch the Java client.

```
<param name="HubURL" value="http://www.host.com:8001/servlet/hub">
<param name="Server" value="mainserver">
<param name="HubAdapter" value="SCExpress">
```

where you must replace www.host.com:8001/servlet/hub with the URL for server hub servlet, and replace mainserver with the name or alias of the ServiceCenter server.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments. Therefore, to connect to the ServiceCenter server through the server hub you would replace the Host and Service parameters with the following:

```
-HubURL=http://www.host.com:8001/servlet/hub -Server=mainserver
-HubAdapter=SCExpress
```

where you must replace www.host.com:8001/servlet/hub with the URL for server hub servlet, and replace mainserver with the name or alias of the ServiceCenter server. You can also specify standalone parameters in the scj.ini file.

Method 2 parameters

Use the parameters in the following table when the Java client is behind a firewall. The port specified in the HTML file that you use to launch the Java client must be open for incoming traffic. For more information, see your system or web administrator.

Parameter	Definition
HubURL	Specifies the URL of the server hub. For example: <paramname="HubURL" value= "http://your_web_server:port_num/hub/servlet/HubServlet"> Setting the HubURL parameter overrides the Java client host and service parameters.
HubAdapter	Specifies the type of server connection to be made. Specify SCExpress.
ClientPort	Specifies the port number on the client machine for the client to use when it connects to the server hub. The port must be available to the client.
Server	Specifies the server name or server hub alias of the ServiceCenter server to connect to that you defined in the server hub servers parameter (for example, mainserver). This parameter is optional if you specify the server hub default parameter.

HTML file example

Replace the Host and Service parameters with the following parameters in the HTML file that you use to launch the Java client.

```
<param name="HubURL" value="http://www.host.com:8001/servlet/hub">
<param name="Server" value="mainserver">
<param name="HubAdapter" value="SCExpress">
```

where you must replace www.host.com:8001/servlet/hub with the URL for server hub servlet, and replace mainserver with the name or alias of the ServiceCenter server.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments. Therefore, to connect to the ServiceCenter server through the server hub you would replace the Host and Service parameters with the following:

```
-HubURL=http://www.host.com:8001/servlet/hub -Server=mainserver
-HubAdapter=SCExpress -ClientPort=12345
```

where you must replace www.host.com:8001/servlet/hub with the URL for server hub servlet, and replace mainserver with the name or alias of the ServiceCenter server. You can also specify standalone parameters in the scj.ini file.

Direct Connection

The Java client makes a direct connection to the server hub through a specified port. If you choose this type of connection, the web server does not assign a random port to the Java client and the network does not assign a dynamic IP address. Your system administrator must also assign a port in the firewall to the Java client. This is the sequence of events:

- The Java client sends a request to connect using an HTTP connection to the server hub.
- The server hub returns an IP address and port number, using the same HTTP connection.
- The client opens a TCP/IP connection to the server hub.
- The server hub runs transparently as a proxy server.

Figure 4-4 shows the connection to a ServiceCenter server using a server hub. Set the ConnectionType parameter in the Java client HTML file used to launch the Java client. Set the connectport parameter in the servlet engine that acts as a container for the server hub.

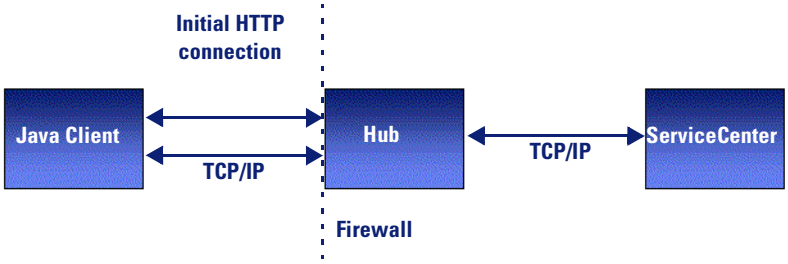


Figure 4-4: .Direct connection

Step 1: Configure the Servlet Engine

The first step in this scenario is to configure the servlet engine to work with the server hub.

1 Configure your servlet engine as shown in *Step 1: Configure the Servlet Engine* on page 78 and perform the following step.

2 If you have a Tomcat server, add the following to the web.xml file:

```
<init-param>
    <param-name>connectport</param-name>
    <param-value>port_number_here</param-value>
</init-param>
```

Warning: The port you specify must be available for incoming connections!

Step 2: Configure the Java Client

The Java client must be aware of the server hub to make the connection to the server hub. Use the parameters in the following table in the HTML file that you use to launch the Java client. You can also pass them on the command line for a standalone application.

Parameter	Definition
HubURL	Specifies the URL of the server hub. For example: http://www.host.com:8001/servlet/hub
HubAdapter	Optional. Specifies the type of server connection to be made. You can specify SCExpress.
ConnectionType	Specifies the type of connection between the server hub and the client. You can specify callback or direct. Choose direct. A direct connection enables the client to initiate the connection to the server hub if the client is running behind a firewall or on a machine that does not have a static IP address.
Server	Specifies the server name or server hub alias of the ServiceCenter server to connect to that you defined in the server hub servers parameter (for example, mainserver). This parameter is optional if you specify the server hub default parameter.

Note: If you use the **HubURL** parameter, it overrides the Java client's **Host** and **Service** parameters. For security reasons, do not set the **Host** and **Service** parameters when you use a server hub.

HTML file example

You can run the Java client as an applet and connect to the ServiceCenter server through the server hub. Replace the **Host** and **Service** parameters with the following parameters in the HTML file that you use to launch the Java client.

```
<param name="HubURL" value="http://www.host.com:8001/servlet/hub">
<param name="Server" value="mainserver">
<param name="ConnectionType" value="direct">
<param name="HubAdapter" value="SCExpress">
```

where you must replace **www.host.com:8001/servlet/hub** with the URL for server hub servlet, and replace **mainserver** with the name or alias of the ServiceCenter server.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments:

```
-HubURL=http://www.host.com:8001/servlet/hub -Server=mainserver
-HubAdapter=SCExpress -ConnectionType=direct
```

where you must replace **www.host.com:8001/servlet/hub** with the URL for server hub servlet, and replace **mainserver** with the name or alias of the ServiceCenter server. You can also specify standalone parameters in the **scj.ini** file.

Direct Connection Without an HTTP Server

You can avoid the HTTP connection altogether by specifying a server with the server hub URL parameter in the HTML file that you use to launch the Java client. For example, you can specify **servername:12345** instead of **http://servername/hub**. The servlet engine must be able to launch servlet zones without an HTTP server. For more information, see your servlet engine documentation. Omitting the HTTP server may improve system security.

The Java client makes a direct connection to the server hub through a specified port. A system administrator must make a port in the firewall available to the Java client if this method is used. This is the sequence of events:

- The Java client sends a request to connect using a TCP/IP connection to the server hub.
- The server hub returns its IP address and port number, using the same TCP/IP connection.
- The client opens a TCP/IP connection to the server hub.
- The server hub runs transparently as a proxy server.

Figure 4-5 shows a direct connection without an HTTP server. The servlet engine must be able to support servlet zones without an HTTP server. Set the ConnectionType parameter in the Java client HTML file used to connect to the server hub. Set the connectport parameter in the servlet engine that acts as a container for the server hub.

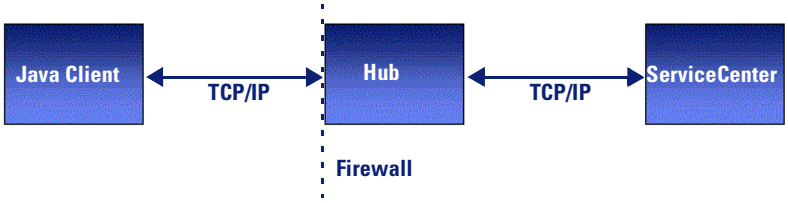


Figure 4-5: Direct connection without an HTTP server

Step1:Configure the Servlet Engine

Follow the steps in *Direct Connection* on page 86 to configure the servlet engine.

Step 2: Configure the Java Client

The Java client must be aware of the server hub to make the connection to the server hub. Use the parameters in the following table in the HTML file that you use to launch the Java client. You can also pass them on the command line for a standalone application.

Parameter	Definition
HubURL	Specifies the URL of the server hub. For example: servername:12345 If you set the HubURL parameter to the URL of the server hub, the server hub will be queried for the host name and port.
HubAdapter	Specifies the type of server connection to be made. Specify SCExpress.

Parameter	Definition
ConnectionType	Specifies the type of connection between the server hub and the client. You can specify callback or direct. Choose direct. A direct connection enables the client to initiate the connection to the server hub if the client is running behind a firewall or on a machine that does not have a static IP address.
Server	Specifies the server name or server hub alias of the ServiceCenter server to connect to that you defined in the server hub servers parameter (for example, mainserver). This parameter is optional if you specify the server hub default parameter.

Note: If you use the HubURL parameter, it overrides the Java client's Host and Service parameters. For security reasons, do not set the Host and Service parameters when you use a server hub.

HTML file example

You can run the Java client as an applet and connect to the ServiceCenter server through the server hub. Replace the Host and Service parameters with the following parameters in the HTML file that you use to launch the Java client.

```
<param name="HubURL" value="servername:12345">
<param name="Server" value="mainserver">
<param name="ConnectionType" value="direct">
<param name="HubAdapter" value="SCExpress">
```

where you must replace servername:12345 with the host name and port of the server hub's direct connection port, and replace mainserver with the name or alias of the ServiceCenter server.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments. Therefore, to connect to the ServiceCenter server through the server hub you would replace the Host and Service parameters with the following:

```
-HubURL=servername:12345 -Server=mainserver
-HubAdapter=SCExpress -ConnectionType=direct
```

where you must replace servername:12345 with the hostname and port of the server hub's direct connection port, and replace mainserver with the name or alias of the ServiceCenter server. You can also specify standalone parameters in the scj.ini file.

Firewall Configurations

If you plan to protect the server hub with a firewall on the client side or the server side (or both), you must configure the firewall properly. You may need to modify the configurations described in this section to support your own environment. Figure 4-6 on page 92 shows a typical firewall configuration with a firewall on each side of the server hub. The firewall between the Java client and the server hub must be open for incoming traffic to enable the client to connect to the server hub. The firewall between the server hub and ServiceCenter must have an open port for inbound ServiceCenter traffic to the IP address used by the server hub.

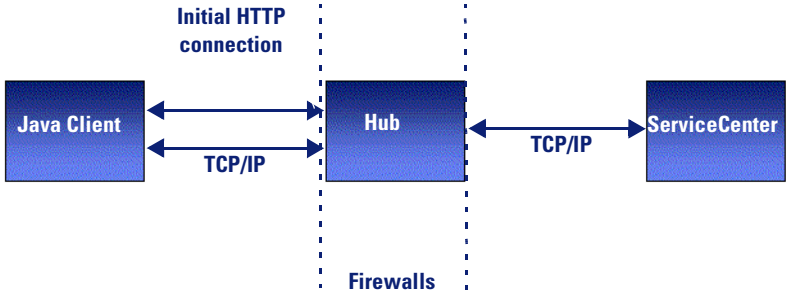


Figure 4-6: Firewall configuration

Java Client Behind a Firewall

If the Java client is behind a firewall, then you must configure the Java client and server hub in one of two ways, depending on how the client IP address is assigned.

IP Address	Java Client Configuration
Static IP address	See *Callback Connection* on page 77.
Network Address Translation (NAT)	See *Direct Connection* on page 86.

Server Hub Behind a Firewall

If the server hub is behind a firewall, then you must configure the Java client in one of two ways, depending on how the client IP address is assigned.

IP Address	Java Client Configuration
Java client with Static IP address	The Java client can make a callback connection to the server hub in this configuration when the following is true: • Specify the Java client ClientPort parameter in the HTML file used to launch the Java client. • The firewall must be open to allow the Java client to connect using the port specified with the ClientPort parameter. For more information, see *Callback Connection* on page 77.
Java client with NAT	The Java client can make a direct connection to the server hub in this configuration when the following is true: • Initialize the server hub with the connectport parameter. • The firewall must allow the server hub to accept incoming connections on the port specified by the connectport parameter. For more information, see *Direct Connection* on page 86.

Example

Follow these steps:

1. Configure your servlet engine following the directions in *Step 1: Configure the Servlet Engine* on page 78.
2. For a Tomcat server, add the following to the web.xml file:

```
<init-param>
    <param-name>connectport</param-name>
    <param-value>port_number_here</param-value>
</init-param>
```

   The port you specify must be available for incoming connections.
3. For a JRun server, add this parameter to the servlet definition

   connectport=available_port_number.

Specify the following parameters in the HTML file that you use to launch the Java client.

Parameter	Definition
HubURL	Specifies the URL of the server hub. For example: http://www.host.com:8001/servlet/hub
HubAdapter	Specifies the type of server connection to be made. Specify SCExpress.

HTML file example

Replace the **Host** and **Service** parameters with the following parameters in the HTML file that you use to launch the Java client.

```
<param name="HubURL" value="http://www.host.com:8001/servlet/hub">
<param name="Server" value="mainserver">
<param name="HubAdapter" value="SCExpress">
```

where you must replace **www.host.com:8001/servlet/hub** with the URL for server hub servlet, and replace **mainserver** with the name or alias of the ServiceCenter server.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments. Therefore, to connect to the ServiceCenter server through the server hub you would replace the **Host** and **Service** parameters with the following:

```
-HubURL=http://www.host.com:8001/servlet/hub -Server=mainserver
-HubAdapter=SCExpress
```

where you must replace **www.host.com:8001/servlet/hub** with the URL for server hub servlet, and replace **mainserver** with the name or alias of the ServiceCenter server. You can also specify standalone parameters in the **scj.ini** file.

Java Client SSL Support

Secure Socket Layer (SSL) provides authentication, encryption, and integrity protection for sensitive data. You can enable SSL support between the Java client and the server hub. Although the Java client connects to the server hub using a **direct** or **callback** connection, it supports SSL only for direct connections. For more information, see *Direct Connection* on page 86.

ServiceCenter SSL support uses a self-signed certificate to authenticate the server. The default cryptographic algorithm for authentication and key exchange is RSA 1024 bit. This value is for key exchange only; the channel is encoded at 128 bit. You can specify key values and certificates in an **scssl.ini** file.

For the current ServiceCenter Java client SSL compatibility information for a specific JRE, see *Peregrine's CenterPoint Web Site* on page 9.

For more information about Java Secure Socket Extensions (JSSE), see the *Java Secure Socket Extension (JSSE) Reference Guide for the Java 2 SDK, Standard Edition, v 1.4*. You can find this guide at *http://java.sun.com/j2se/1.4/docs/guide/security/jsse/JSSERefGuide.html.*

SSL System Requirements

The server hub requires JRE version 1.3.x and Apache Tomcat version 4.1.x. The client requires JRE version 1.3.x

Creating SSL Support

There are a few basic tasks to enable SSL support.

Step 1 Enable SSL support on the server. See *Server-Side SSL Support*.

Step 2 Enable SSL support on the client. See *Client-Side SSL Support* on page 98.

Step 3 Verify that you enabled SSL successfully. See *Verify SSL for Server and Clients* on page 101.

Server-Side SSL Support

Before you begin:

1. Review the information in this chapter about the client, server, and server hub relationship
2. Install the server hub using the direct connection method. See *Direct Connection* on page 86 or *Direct Connection Without an HTTP Server* on page 89.
3. Obtain your key:
   - Use the default encryption key shipped with server hub and proceed to *Using a default encryption key*.
   - Use a private encryption key and proceed to *Using a private encryption key* on page 96.

Using a default encryption key

Follow these steps to enable SSL on the server:

1. Open this file with a text editor:

   TOMCAT_HOME>/webapps/<SERVER_HUB_NAME>/WEB-INF/web.xml

2. Find the last instance of the </init-param> tag.
3. On the next line, insert this new <init-param> tag:

```
<init-param>
    <param-name>sslconnectport</param-name>
    <param-value>portvalue</param-value>
</init-param>
```

4. Save and close the file.
5. Follow the instructions in *Client-Side SSL Support* on page 98.

Using a private encryption key

Follow these steps to enable SSL on the server:

1. Open this file with a text editor:

   TOMCAT_HOME/webapps/<SERVER_HUB_NAME>/WEB-INF/web.xml

2. Find the last instance of the </init-param> tag.

3 On the next line, insert this new <init-param> tag:

```
<init-param>
    <param-name>sslconnectport</param-name>
    <param-value>portvalue</param-value>
</init-param>
<init-param>
    <param-name>scjssl</param-name>
    <param-value>sslserver.ini</param-value>
</init-param>
```

4 Save and close the file.

5 Create a new file with a text editor.

6 Type these two lines:

```
Keystore=peregrine.key
Keystorepass=password
```

7 Save the file as sslserver.ini in this directory:

TOMCAT_HOME/webapps/<SERVER_HUB_NAME>/WEB-INF

8 Close the file.

9 Create a key file that contains the private key. Name the file peregrine.key.

Note: Each JRE provides an encryption key tool that creates a key file with an encrypted key. For more information, see your JRE documentation.

10 If you run JRE 1.4.*x*, your server-side installation is complete. Proceed to the instructions in *Client-Side SSL Support* on page 98. If you run JRE 1.3.*x*, complete the steps in *Retrieve SSL libraries (JRE 1.3.x only)* first.

Retrieve SSL libraries (JRE 1.3.*x* only)

Follow these steps to download SSL libraries:

1 Download the Sun SSL libraries (JSSE) from one of the following:

- *http://java.sun.com/products/jsse/*
- *http://support.peregrine.com*

2 Copy the following library files from the download:

- jsse.jar
- jnet.jar
- jcert.jar

3 Paste these three files into your JRE 1.3 /.../lib/ext directory.

4 Stop and restart the application server.

Warning: Stopping and restarting he application server stops all Java Client user sessions.

Client-Side SSL Support

The server supports two types of client connections, depending on the key.

- If the client connects to the server with the default encryption key shipped with server hub, see *Connecting a Windows Java client with a default key* or *Connecting a browser-based Java client with a default key* on page 99.
- If the client connects to the server with a private encryption key, see *Connecting a Windows Java client with a private key* on page 99 or *Connecting a browser-based Java client with a private key* on page 100.

Connecting a Windows Java client with a default key

Follow these steps:

1 From the **Windows Start** menu, **select Programs** > **Peregrine ServiceCenter**

2 Right-click **Java Client**, then select **Properties**.

3 On the **Shortcut** tab, in the **Target** field, type the following at the end of the target field:

-sslenabled=true

For example:

```
C:\jdk1.3.1_07\jre\bin\java.exe -classpath jbird.jar
com.peregrine.sc.client.ClientApplication
-HubURL=http://<servername>:8080/hub/hub  -Server=mainserver
-ConnectionType=direct -sslenabled=true
```

4 Click **Apply**.

Windows NT:

You cannot complete step 2. Create a shortcut for the Java client on the desktop and right-click the shortcut to edit the target field.

Connecting a browser-based Java client with a default key

Follow these steps:

1. From the **Windows Start** menu, select **Programs** > **Peregrine ServiceCenter.**
2. Right-click the browser-based **Java Client** and select **Properties.**
3. On the **Shortcut** tab, look for the **Target** field.
4. Open the //Peregrine/ServiceCenter/java/scjava.htm file with a text editor.
5. Add the following:

   ```
   <param name= “sslenabled” value=”true”>
   ```

6. Save and close the file.

   Windows NT:

   You cannot complete step 2 on page 99. Create a shortcut for the Java client on the desktop and right-click the shortcut to edit the target field.

Note: For more information, see *Java Client .htm Files* on page 56.

Connecting a Windows Java client with a private key

1. From the **Windows Start** menu, select **Programs** > **Peregrine ServiceCenter.**
2. Right-click **Java Client**, then select **Properties**.
3. On the **Shortcut** tab, in the **Target** field, type the following at the end of the target field:

   ```
   -sslenabled=true -scjssl=sslclient.ini
   ```

   For example:

   ```
   C:\jdk1.3.1_07\jre\bin\java.exe -classpath jbird.jar
   com.peregrine.sc.client.ClientApplication
   -HubURL=http://<servername>:8080/hub/hub  -Server=mainserver
   -ConnectionType=direct -sslenabled=true -scjssl=sslclient.ini
   ```

4. Click **Apply**.

Windows NT:

You cannot complete step 2. Create a shortcut for the Java client on the desktop and right-click the shortcut to edit the target field.

Connecting a browser-based Java client with a private key

1. From the **Windows Start** menu, select **Programs** > **Peregrine ServiceCenter**.
2. Right-click the browser-based **Java Client** and select **Properties**.
3. On the **Shortcut** tab, look at the **Target** field.

Windows NT:

You cannot right-click the Java client. Create a shortcut for the Java client on the desktop and right-click the shortcut to edit the target field.

1. Open the //Peregrine/ServiceCenter/java/scjava.htm file with a text editor.
2. Add the following:

```
<param name= “sslenabled” value=”true”>
<param name= “scjssl” value=”sslclient.ini”>
```

3. Save and close the file.

   Note: For more information, see *Java Client .htm Files* on page 56.

4. Complete the steps in the next section, *Creating a truststore file*.

Creating a truststore file

Your JRE environment provides a key tool. Use the key tool to create a truststore file named scserver.cert. Ensure that the file contains your certificate.

To create a truststore file:

1. Use a text editor to create a new file.
2. Type the following in that file:

```
Truststore=sslclient.ini
Truststorepass=password
```

3. Save the file as sslclient.ini in this directory:

   TOMCAT_HOME/webapps/<SERVER_HUB_NAME>/WEB-INF

4. Close the file.

Verify SSL for Server and Clients

To verify SSL:

1. From the **Windows Start** menu, select **Programs** > **Peregrine ServiceCenter** > **Java Client.**
2. When the Java client appears, look in the lower right corner of the Java Client window for the lock and key icon. If the icon appears, SSL is enabled for that particular client.
3. Log in as falcon (or any other valid operator).
4. On the Server, open the Java Server servlet log file.

   The location of the log file varies, depending on which Java Server servlet runner you are using and how it is configured. For more information, see *Troubleshooting* on page 107.
5. Search the log file for the following:

   secure socket
6. Read the text in the secure socket log entry. The text should indicate that the login in a previous step was successful and that SSL is enabled.

Default Cipher Suites

A cipher suite is a combination of cryptographic parameters that define the security algorithms and key sizes used for authentication, key agreement, encryption, and integrity protection. The SSL libraries provide support for cipher suite negotiation, which is part of the SSL handshaking that initiates or verifies secure communications. ServiceCenter SSL support includes the following cipher suites (listed in preference order).

SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

CHAPTER 5

ServiceInfo Universal

ServiceInfo Universal (SIU) is a servlet that displays and refreshes a ServiceCenter form. You can accomplish the same task without using servlets by using the Publish and Subscribe feature. For more information, see Publish and Subscribe in the *System Tailoring, Volume 1* guide.

Read this chapter for more information about:

SIU Servlet Requirements

There are four components that you work with to install and run an SIU servlet:

- A web server or a standalone servlet engine
- Servlet engine
- Java client
- ServiceCenter server

The installation installs the SIU servlet file as part of the standard Java client installation. It consists of a single Java JAR file: **serverhb.jar**. The default path to this file is \\Peregrine\ServiceCenter\ java**serverhb.jar**. To configure the SIU connection, there are three tasks:

- Configure the web server for servlet zones or the servlet engine you use must be able to launch a servlet without an HTTP server.
- Configure the servlet engine to include the **serverhb.jar** file.
- Configure the ServiceCenter Java client to connect to the SIU.

The SIU servlet package that is passed to the servlet engine is **com.peregrine.hub.sc.SIUServlet**. This package name is case sensitive.

Using Java Servlets

The SIU is a Java servlet. Java servlets are server side Java programs that can generate dynamic content. They support a request/response model that is commonly used in servers. They are more efficient than CGI programs, which start a new process each time they are invoked. From the client side, you can access a Java servlet in the same way as a standard CGI script. Java servlets are platform-independent as well as faster and more secure than CGI scripts.

SIU Parameters

The following are the SIU connection parameters and apply to all servlet engines. These parameters apply to the SIU servlet and are set when configuring the servlet engine.

The following table lists SIU connection parameters that apply to the SIU servlet. You can set these parameters when you configure the servlet engine. These parameters are case sensitive and you must include all of them or the SIU will not run.

Parameter	Description
Heartbeat	Specifies how often the SIU queries the ServiceCenter server for updates in seconds. Specify a numeric value in seconds. For example: `Heartbeat=25`
Timeout	Specifies how long to wait for the ServiceCenter server to respond before exiting. Specify a numeric value in seconds. For example: `Timeout=25`
Verbose	Specifies whether the servlet should dump error information. Valid values are True and False. For example: `Verbose=25`

For an example of how to configure a servlet engine, see *Tomcat example* on page 78 or *JRun 3.1 example* on page 81.

HTML File Parameters

The Java client must be aware of the SIU to make the connection to the SIU. The following table lists the connection parameters that you can insert in the HTML file that you use to launch the Java client (**scjava.htm**, **scjavamac.htm** or **scjavaplugin.htm**). You can also pass parameters on the command line for a standalone Java client.

Note: Unlike the server hub, the SIU servlet requires that you define the Host and Service parameters in the HTML file that you use to connect to the SIU.

Parameter	Description
HubURL	Specifies the URL of the SIU. For example: http://www.host.com:8001/servlet/siu
SCAdapter	Specifies the type of server connection to be made. You must use SCExpress.
SIU	Specify the name of the form to display at connection time. For example: pm.status
Host	Specifies the host IP address of the ServiceCenter server. This address must be accessible by user machines.
Service	Specifies the ServiceCenter server port.

HTML file example

Specify the following parameters in the HTML file that you use to launch the Java client. These parameters enable the Java client to run as an applet and connect to the ServiceCenter server through the SIU.

```
<param name="HubURL" value="http://www.host.com:8001/servlet/siu">
<param name="SCAdapter" value="SCExpress">
<param name="siu" value="form_to_display">
<param name="Host" value="ServiceCenter_host_name">
<param name="Service" value="form_to_display">
```

where you replace www.host.com:8001/servlet/siu with the URL for the SIU servlet installed on your system.

Standalone example

When you run the Java client in standalone mode, you can pass parameters to the client as command line arguments.

```
-HubURL=http://www.host.com:8001/servlet/siu -siu=display_form
-HubAdapter=SCExpress -Host:host.company.com -Service:12670
```

You can also specify standalone parameters in the scj.ini file.

6 Troubleshooting

CHAPTER

It is important to solve problems as they arise. Some are more common than others. Read this chapter for trouble shooting information about:

Windows Systems

The problems in this section occur on a Windows operating system.

Running the Java client in a browser

If you have problems running the Java client in a browser under Windows. Peregrine Systems suggests that you download the latest JVM from Microsoft's Web site.

Client/Server dependency

The Java client can be upgraded independently of ServiceCenter. If you have problems with the Java client, you can upgrade to the latest version without upgrading your ServiceCenter server or applications.

- ServiceCenter version 3.0 SP3 Java client is compatible with all 3.0 servers.
- ServiceCenter version 4.0 Java client is compatible with all 3.0 servers and the 4.0 server.
- ServiceCenter version 5.0 Java client is compatible with all 3.0, 4.0, 5.0, and 5.1 servers.

Cannot launch multiple Java client sessions

Microsoft Internet Explorer 5.0 is unable to launch more than one browser client at a time. To launch multiple browser clients, you must use Internet Explorer 5.5 or a later release.

Applet configuration

To help you troubleshoot problems with execution, the client generates log messages that you can view in the browser's Java Console.

- To view log messages with Internet Explorer, you must turn on Java logging.
  - From the **Tools** menu, select **Internet Options.**
  - Click the **Advanced** tab.
  - Scroll down to find **Microsoft VM.**
  - Select **Java logging enabled** and **Java console enabled**.
  - Click **OK**. Restart your browser.
  - From the **Tools** menu, select **Java Console**.

- To view log messages with Netscape Navigator 4.5, select the **Java Console** option on the **Communicator->Tools** menu.

For browser compatibility information, see *Peregrine's CenterPoint Web Site* on page 9.

Standalone Application Configuration

To troubleshoot problems with a standalone application, the client generates a file named **scj.log**. From the **Help** menu, select **About** on the Java client menu bar for the location of the log file on your system.

If you are unable to start the Java client, ensure that you have a client connection to the ServiceCenter server. You can use a standard Windows client to validate the connection to the server.

MaxChars Parameter

The MaxChars parameter that you can set in the Forms Designer properties window, is enforced by the Java client. The default setting is 0. If you change this setting, the result may be text strings cut off in the objects that use this parameter.

The solution is to change the setting in the Properties window to a higher value, or to return it to the default setting of 0.

Unix Systems

The problems in this section occur on a Unix operating system.

Copy and Paste (all)

On Unix/CDE machines there are typically two selection buffers; PRIMARY and CLIPBOARD. The PRIMARY buffer normally receives any highlighted text, which you can paste into other applications using the middle mouse button. Unfortunately, Java currently supports only the CLIPBOARD buffer for cut and paste operations. To work around this problem, you may have to use an intermediary application (such as dtterm or xclipboard) that can access both buffers.

To copy text from xterm into the Java client:

1. Select the text in the xterm window.
2. Use the middle mouse button to paste the text into a dtterm window.
3. Select the text in the dtterm window. From the **Edit** menu, select **Copy**.
4. Insert the cursor in the desired text component in the Java client. From the **Edit** menu, select **Paste**.

To copy text from the Java client into xterm (the reverse):

1. Select the text in the Java client. From the **Edit** menu, select **Copy**.
2. Insert the cursor in the dtterm window. From the **Edit** menu, select **Paste**.

To copy text from dtterm into xterm:

1. Select the text in the dtterm window.
2. Use the middle mouse button to paste the text into the xterm window.

HP-UX 11 and Netscape

If you use Netscape on a HP-UX 11 systems, start your browser with the -visual Default switch. For example:

```
netscape -visual Default
```

Linux

The copy and paste functions do not work if the application does not share the same clipboard as the Java client.

Client (all)

If you want to run a browser-based Java client from a browser on a Unix system, you must ensure that you can start the Java console. If the console does not appear, the Java VM is not configured properly and is not accessible.

See your browser documentation to ensure that the Java classes are accessible. In some cases this means resetting the CLASSPATH environment. For Netscape Navigator, set the MOZILLA_HOME environment variable to point to the base (directory) location of the browser.

If you want to run a standalone Java client, some systems work better with different JREs. If the Java client throws an exception or fails, install a later version of the JRE, or revert to a previous version of the JRE.

SUN OS

An exception occurs if you attempt to start the Java Client as a user and you did not started the current X Window session.

The Java Client user must own the current X Window session. You can use the xhost + command to disable xhost security, which permits users from any system to log in and control the X environment; however, this is a less secure method.

Macintosh Systems

cannot connect null:null

If you cannot launch the Java client as a standalone application or you receive the error message cannot connect null:null, store the scj.ini file in the preferences directory and restart the client.

MRJ 2.2.4

The problem is that you can tab only between the browser's address field and the first field of the displayed form in the Java Client. This is a known problem with Internet Explorer Java support on a Macintosh system. The browser processes all tab key events without presenting them to the Java applet running from the browser. This is true for the ServiceCenter Java client and all other Java applets.

Server Hub

The server hub generates basic diagnostic messages that are useful to track problems that occur. Examine the log file to troubleshoot configuration issues. You can email the log file to Peregrine Systems, Inc. support with a specific description of the events that reproduce the problem.

The location of the log file varies depending on the servlet engine you are using and its configuration. For example, if you use the JRun server with its default log file settings, the log file resides in:

JRun_Home/jsm-default/logs/stdout.log

where JRun_Home is the root directory for the JRun installation.

UseGetMethod Parameter

Specify this parameter to connect to the server hub only for certain combinations of web browsers and servlet engines. Do not use it unless you have difficulty connecting to the servlet from a browser, and you know the servlet is configured properly. The UseGetMethod parameter has a value of true or false. The default value is false.

When the UseGetMethod parameter has a value of true, the client attempts to connect to the servlet using the HTTP GET method. Otherwise it uses the HTTP POST method. The syntax for the parameter is:

```
<param name="UseGetMethod" value="true">
```

HTML files

Do not insert back slashes (\) in an HTTP URL. Use only forward slashes (/). For example, this expression is valid:

```
<param name="HubURL" value="http://localhost:8001/servlet/hub">
```

This expression is invalid and will generate a connection error message:

```
<param name="HubURL" value="http://localhost:8001\servlet\hub">
```

All Systems

Drag-and-drop

The Java client does not support drag-and-drop operations. Attachments must be placed using menu commands.

CHAPTER 7

Accessibility Specifications

When Section 508 became an addendum to the Rehabilitation Act of 1973, it required United States federal agencies to make electronic and information technology usable by anyone with a disability. ServiceCenter meets this requirement through Section 508 compliance in the Java client.

The Java Swing library is a graphical user interface development tool that is part of the Java 2 Platform, Standard Edition (J2SE). It provides direct support for accessibility in the final software product. When software contains these accessibility features, assistive technology vendors can customize the look-and-feel of the software to support non-visual presentation with audio or other devices. This means that Java client users can configure the interface to meet their special needs when they apply third-party assistive technology tools.

Read this chapter for information about:

Section 508 Compliance Features

The following general features support Section 508 compliance.

- Most text fields and objects in the Java client have assigned name and description fields.
- Forms Designer supports manually assigning name or description fields to the remaining text fields and objects. This is also available in the Windows client for compatibility.

  For more information about Forms Designer, see *System Tailoring, Volume 1*.
- Read-only and other mandatory components have character strings appended to the Accessible Name field to ensure that you can use third-party assistive technology to read these strings and components aloud.
- You can save your preferences for keyboard access, flashing, fonts, look-and-feel, and color schemes in the Java client initialization file.
- You have a command line option to enable preferences defined for a specific user before you log in to the Java client.

Keyboard Features

These features enhance keyboard access to the Java client. You can:

- Specify optional keyboard access using the TAB key into read-only fields, the status bar, Message Boxes, and to enable customized tabbing through table fields.
- Select a menu option to enable or disable keyboard access to all fields.
- Select a menu option to override the default tab order with a top-down instead of left-to-right tab order.

Viewing Preferences

These features improve viewing the Java client. You can:

- Choose a menu option to enable or disable flashing components, such as the Activity Icon.
- Access a font selection dialog to change font family, point size, and emphasis.

- Choose a menu option to change the look and feel of the Java client.
- Access a color scheme definition and selection dialog.

Assistive Technology Tools

The Java client enhancement for Section 508 has some embedded features that enable you to use it with assistive technologies and tools developed by third-part vendors. These are standard accessibility components that expand functionality for disabled users.

Third-Party User Tools

There are different assistive technology tool vendors that produce software and hardware designed to meet the needs of users. Hardware device, such as braille workstations and large key keyboards, and software products, such as screen magnification and speech recognition software, can improve user interaction with any software. Some software tools can read the contents of the current window, such as JAWS by Freedom Scientific. Other voice recognition tools can accept dictation from the user and translate that into usable text. Two examples are Via Voice by IBM and Dragon Naturally Speaking by ScanSoft.

Development Tools

Java developers use these tools to enable the software, the operating system, and third-party accessibility products to interact successfully.

Java Accessibility API

The Java Accessibility API is an application development interface that provides information and content to assistive technologies, such as Jaws for Windows.

Java Accessibility Utility Package

The Java Accessibility Utility package enables assistive technology vendors to locate and access all components, such as fields, buttons, check boxes, and radio buttons, inside a Java application that runs in a JVM environment.

Java Accessibility Bridge

The Java Accessibility bridge connects software that runs in the JVM with software running on a native platform. It is unique for each platform. This bridge enables the Java client to take advantage of native operating system features that promote accessibility and to communicate with third-party assistive tools.

Note: This bridge is currently available only for the Windows operating system.

Setting Editing Options

You can set preferences for disabling animation and gaining access to all components (such as, fields, buttons, check boxes, and radio buttons), including read-only fields.

To set accessibility and animation preferences:

1. From the **Edit** menu, choose **Preferences** > **Accessibility** > **Access all Fields** to enable text reader assistive technologies, such as JAWS, to read through the components of any window (fields, buttons, check boxes, or radio buttons) within ServiceCenter.
2. From the **Edit** menu, choose **Preferences** > **Accessibility** > **Ignore Server Tab Order** to tab from left to right in the window. If you check Ignore Server Tab Order, tabbing is top to bottom.
3. From the **Edit** menu, choose **Preferences** > **Accessibility** > **Disable Animation** to disable the blinking status light and messages within the status bar.

Setting Viewing Options

Pluggable look-and-feel architecture enables the Java client to emulate the look-and-feel of Windows and Unix Motif operating systems, and to provide optional color schemes. This feature also enables assistive technology vendors to provide custom look-and-feel for audio presentation or special hardware devices.

To set the look-and-feel of the interface:

- From the **Edit** menu, choose **Preferences** >**Look and Feel**. You can choose **Metal**, **CDE/Motif**, or **Windows** to change the overall look of the Java Client interface.

To set color scheme:

From the **Edit** menu, choose **Preferences** > **Color Scheme**. Figure 7-1 shows two drop-down menus that enable you to choose a different color scheme and to see how that color scheme affects colors assigned to different objects in the interface.

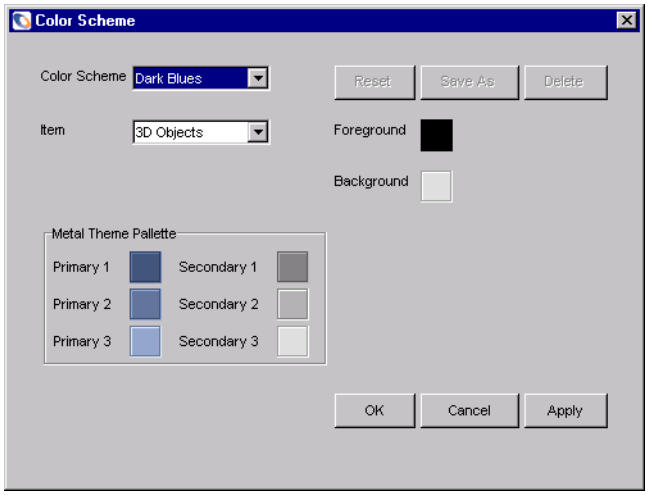


Figure 7-1: Color Schemes

Section 508 Compliance Issues

There a few limitations in Java Client version 5.1 that may impact your choices. Some potential features are not available yet and there are some known issues that relate to third-party accessibility tools.

Known Issues

There are some issues that impact Java client users who want to take advantage of the accessibility features.

- The JAWS version 4.5 does not read the available data in tables.
- JAWS reads only the topmost node of the Explorer tree, and does not recognize selection changes to other nodes.
- A few forms contain fields or object names that cannot be assigned an accessible name. You can assign them manually using Forms Designer.
- The Forms Designer utility is not Section 508 compliant. An administrator should make required forms modification to support the requirements of the target user.

Deferred Features

A few features are unavailable in this version of Java client accessibility.

- Accessible text in marquee components.
- Accessible names and descriptions for graphics.
- Accessible alternatives for chart components.

Index

D

E

F

H

I

J

R

S

T

U

V

W

X


Peregrine
SYSTEMS