建立及使用 IBM Lotus Sametime 即時傳訊技術所支援的即時連接埠監視應用程式

等級：中級

James Dewan (jdewan@us.ibm.com)，資深支援經理，IBM

Jennifer Wales (jennifer_wales@us.ibm.com)，經過 IBM 認證的 IT 諮詢專員，IBM

2008 年 5 月 20 日

瞭解如何建立連接埠檢查應用程式，以搭配使用 IBM® Lotus® Sametime® 即時通知環境中服務中斷的情形。Lotus Sametime 工具集功能強大，可建立運用 Lotus Sametime 技術的自訂應用程式。

本文會說明 Healthcheck 連接埠檢查程式的開發過程。當初是爲了解決客戶端的某些環境問題建立此應用程式，隨後還結合了 IBM Lotus Sametime 7.x 版及以上的功能。雖然此應用程式有助於解除若干問題，但直到轉型成 Lotus Sametime bot 程式後才真正發揮其實力。Healthcheck 及 Lotus Sametime 整合後，成爲全年無休的伺服器監視工具，可即時通知一或多位系統管理員服務中斷情形。這個 Pure Java™ 應用程式可在任何支援軟體開發套件 (SDK) 1.4.2 的平台上執行。

本文所探討的範疇如下：

- 應用程式雛形的設計背景
- 原始應用程式的程式碼範例
- 發揮 Lotus Sametime 即時傳訊 (IM) 的功能，讓此應用程式達到即時效果
- 此應用程式的效用
- 說明如何納入 Lotus Sametime 的全新應用程式碼

環境及問題範圍

Healthcheck 原本是診斷工具，可用來判斷應用程式是否在特定連接埠上回應。網路管理者一向習於使用 Ping 公用程式判斷主機的連線是否正常。如果使用者有應用程式方面的問題，其中一個疑難排解方式就是 Ping 該主機，以確認該用戶端是否可正常連線到執行應用程式的主機。

Healthcheck 的功能更強，可讓您 Ping 應用程式正在接聽的特定連接埠。您可使用 Healthcheck 來進行單次查詢，疑難排解連線問題或持續監視應用程式。Healthcheck 最出色的功能是可從文字檔擷取輸入項目。只要瞭解商業應用程式所使用的連接埠，您便可將 Healthcheck 當作監視各種伺服器的工具，以及連線問題的特定疑難排解工具。

Healthcheck 適用的環境包括基礎架構中的各種 Lotus 伺服器，如 IBM Lotus Domino® 郵件、應用程式、LDAP 及 SMTP 伺服器；Lotus Sametime 即時傳訊、會議、Multiplexer (MUX) 與 SIP 伺服器，以及 IBM Lotus QuickPlace® 伺服器。

Healthcheck 應用程式的基本設計和必要項目

Healthcheck 應用程式起初強調的是提供顧客方便使用輸入檔的方法、匯入所有需要監視的資源，以及提供單次可用性報告或通知網路斷線的持續查詢。

必要項目

Healthcheck 最基本的需求之一就是盡量減輕重量以方便攜帶，這也意味著以下三點：

- **可在桌上型電腦至大型伺服器的平台上執行該應用程式。**開關遠端主機/伺服器 Socket 的功能主要是網路作業，而且不需要針對執行該應用程式的影像進行大量處理工作。
- **用戶端只需安裝少數軟體。**Healthcheck 的唯一軟體要件是在路徑中安裝 Java 執行時期環境 (JRE) 1.4.2。此外，任何主機、應用程式或伺服器類型，一律無須產品外掛程式，這樣一來，Healthcheck 即可限制在網路 Socket 檢查，不需要安裝產品元件來配合受監視伺服器或主機的類型。
- **可在任何平台上執行。**Healthcheck 是以 Java 撰寫的跨平台應用程式，幾乎可在任何平台上執行。Java 的 JRE 1.4.2 可提供許多平台的原生支援，包括 Microsoft Windows、UNIX、Linux™ 及主機型系統，不需要在監視主機上安裝特定平台的軟體元件。

Healthcheck 屬於多功能工具，不僅是可安裝在任何網路位置，還可從同一位置監視無數使用網路連線的應用程式及資源。

顯示伺服器狀態

Healthcheck 納入 Lotus Sametime 功能之前，只有一種方法可以通知使用者所要求的查詢狀態。

Healthcheck 的一次性呼叫功能會顯示視窗，列出所要求主機或連接埠的 Socket 連線清單。若無法連線，該主機或連接埠就會連同 Socket 開放呼叫所產生的錯誤碼一起顯示。

清單 1 和 2 說明如何使用 JPanel Java 物件匯入本端 Display Window，以及如何控制主機/連接埠連線失敗的蹦現視窗。

清單 1. 匯入本端顯示畫面的程式碼

```
void PrintWindow()
{
// Does a local pop-up window if -service not specified on input of all errors found
          try {
                    DisplayWindow dw=new DisplayWindow(outputfile,Errors,error_index);
                    dw.setLocationRelativeTo(null);
                    dw.setVisible(true);
          }
          catch (Exception e)
          {Printnow("exception from DisplayWindow");}

}
```

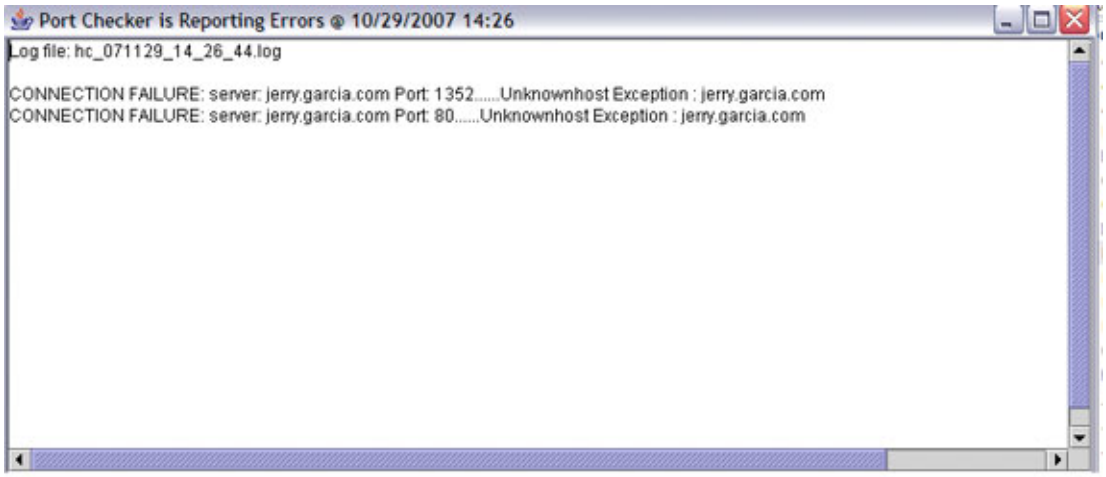
清單 2. 控制蹦現視窗的程式碼

```
import java.awt.*;
import javax.swing.*;
import java.util.Calendar;

public class DisplayWindow extends JFrame {
    //============================================== instance variables
   JTextArea _resultArea = new JTextArea(20, 80);

    //================================================== constructor
    public DisplayWindow(String logname, String Errors[], int error_index) {



          Calendar c = Calendar.getInstance();

        //...Set textarea's initial text, scrolling, and border.
          _resultArea.append("Log file:" + logname + ".log¥n¥n");
          for (int i = 0;i<error_index;i++)
          {
          _resultArea.append(Errors[i]+"¥n");
          }
        //_resultArea.setText("Enter more text to see scrollbars");
        JScrollPane scrollingArea = new JScrollPane(_resultArea);

        //...Get the content pane, set layout, add to center
        JPanel content = new JPanel();
        content.setLayout(new BorderLayout());
        content.add(scrollingArea, BorderLayout.CENTER);

        //...Set window characteristics.
        this.setContentPane(content);
          String ampm;
          if (c.get(Calendar.AM_PM) == 1)
                    ampm=new String("PM");
          else
                    ampm=new String("AM");

        this.setTitle("Port Checker is Reporting Errors @ "+ c.get(Calendar.MONTH)+"/"+
        c.get(Calendar.DAY_OF_MONTH)+"/"+c.get(Calendar.YEAR)+" "+
```

```
        c.get(Calendar.HOUR_OF_DAY)+":"+c.get(Calendar.MINUTE) );
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        this.pack();


    }
}
```

圖 1 是所產生的錯誤畫面。

圖 1. 列出 Healthcheck 錯誤的本端蹦現視窗



將 Healthcheck 當作 Microsoft Windows 服務或 AIX 常駐程式來執行

爲使 Heathcheck 進一步主動發揮功能，新增服務選項後，使用者不必在本端登入監視主機，應用程式也能執行。此外，該服務選項可讓 Healthcheck 自動重新啓動，例如需要重新啓動監視機器進行維護或安裝修補程式。Microsoft® Windows® 服務啓用及 AIX® 常駐程式支援是透過 Tanuki Software 所開發的 Java Service Wrapper 開放程式碼專案來達成。此開放原始碼工具可提供封套，透過 Microsoft Windows Services 或 AIX 常駐程式指令控制 Healthcheck。

輸入規格

爲了提供輸入檔及方便將資源歸類，該內容檔的格式不需要重複指定每個列出主機的相同連接埠。輸入檔 [Ports]（連接埠）區段下的連接埠設定，提供了指定特定伺服器類別的連接埠位置：[Domino]、[Sametime]、[MUX]、[SIP]、[QuickPlace]。這個位置可讓 Healthcheck 充分檢查每個類型及使用該連接埠清單，以檢查每個所需連接埠。

若特定輸入或伺服器使用了無法輕鬆歸類的連接埠，即可使用統稱 [Hosts] 的種類來指定個別主機和連接埠。圖 2 是輸入檔範例。

圖 2. healthcheck.properties 輸入檔的範例

```
[Ports] <- Case sensative entries in this section
Sametime_Ports=1352,80,8081,8082,1533,999
Domino_Ports=1352,80
Quickplace_Ports=1352,80,389
Mux_Ports=1533,8082
Sip_Ports=5060,5061
[Domino]
domino1.test.com
domino2.test.com
domino3.test.com
domino4.test.com
[Sametime]
Sametime01.test.com
[Sip]
SIP01.test.com
[Quickplace] <-- if section is empty, then no entries
[Hosts]
Host.test.com;80,1024
Host2.test.com;80,81
Host3.test.com;1033,1024
Host4.test.com;80,1024
[Options] <- Case sensative entries in this section
Check_Interval=60
Sametime_Notification_Users=User1 Lastname/XXX/XXX,User2 Lastname/XXX/XXX,User3 Lastname/XXX/XXX
Sametime_Notification_Server=sametime.test.com
Sametime_login_user=login_user_name
Sametime_login_pw=your_pw
```

連接埠監視功能的實作

誠如前述，Healthcheck 的設計目的之一，是爲了建立跨程試碼的平台和應用程式，同時說明主機/連接埠組合無法連線的原因，使用 Java Socket 類別 java.net.socket 後已達成了這個目標。清單 3 說明的程式碼範例，是用來監視各主機或連接埠並追蹤異常狀況。

清單 3. 用來檢查輸入檔所列之 Lotus Sametime 伺服器的程式碼範例

```
void CheckSametime() {
int Ports[]=new int[100];
int token_count=1;

String port_ini_setting=FindPortInfo(SAMETIME_PORTS);

if (port_ini_setting.equals(""))
          {
Ports[0]=1352;
Ports[1]=80;
Ports[2]=1533;
Ports[3]=8081;
Ports[4]=8082;
token_count=5;
```

```
          }
else
          {

StringTokenizer st = new StringTokenizer(port_ini_setting,",");

        while (st.hasMoreTokens())
        {
          try {
                    Ports[token_count-1]=Integer.parseInt(st.nextToken());
                    // catch a NumberFormatException
                    token_count++;
          }
          catch (Exception port)
          {
                    Printnow("Port value is not a valid #");
          }

}
          } // end of if

for (int index=0;Sametime[index]!=null;index++)
          {
          Printnow("¥nServer:" + Sametime[index]);




          for (int ports_to_check=0; ports_to_check < token_count-1; ports_to_check++)
          {
          try
          {
            Socket con = new Socket(Sametime[index],Ports[ports_to_check]);
          if (con !=null)
          {
          Printnow("Connection was successful for server:" + Sametime[index] + " Port:"
                    + Ports[ports_to_check] );
                    con.close();
          }
```

```
          }
          catch (UnknownHostException unoe)
          {
          //Printnow("CONNECTION FAILURE:server:Unknown host " + Quickplace[index] +
          " Port:" + Ports[ports_to_check] + "     );
          Printnow("CONNECTION FAILURE:server:" + Sametime[index] + " Port:"
          + Ports[ports_to_check] + "......Unknownhost Exception : " + unoe.getMessage() );
          }
          catch (IOException ioe)
          {
          //Printnow("CONNECTION FAILURE:server:IOException " + Quickplace[index]
          + " Port:" + Ports[ports_to_check] );
          Printnow("CONNECTION FAILURE:server:" + Sametime[index] + " Port:"
          + Ports[ports_to_check] + "......IOException : " + ioe.getMessage() );
          }
          catch(SecurityException sece)
                      {

                     Printnow("CONNECTION FAILURE:server:" + Sametime[index] + " Port:"
                     + Ports[ports_to_check] + "......Security Exception : "
                     + sece.getMessage() );
                     //Printnow("exception  " + ioe.getMessage());

                     }

   }
          //token_count=1;
   }
}
```

如果是透過建立新 Socket 物件來建立和開啓 Socket 連線，該測試就算成功，隨即 Socket 也會關閉。然而，如果無法實例化新的 Socket 物件，異常訊息擲出後就會遭應用程式攔截。如果是 Socket 類別，無法建立 Socket 物件時擲出的異常即可分爲三種，如表 1 所示。

表 1. 異常錯誤類型

異常	特徵	意義/用途
UnknownHostException	無法透過 DNS 聯繫特定主機。	• 若網路中斷影響 DNS 功能或是一般網路問

		題，則至關重要。 • 可識別因爲網路問題而無法連接主機的相關服務中斷情形。 • 這些錯誤通常不是特定應用程式的問題。
IOException	可以連線主機，但無法開啓 Socket 連線至特定連接埠。	• IOExceptions 通常指向不是接聽已配置之連接埠的伺服器/應用程式。 • 可以連線主機，但無法連到特定連接埠。
SecurityException	有安全管理程式，但 checkConnect 方法不允許該作業。	• 基於安全因素而無法連線應用程式/連接埠。

使用 Lotus Sametime 強化 Healthcheck：即時通訊的威力

Healthcheck 的初步成果和測試令人雀躍。此工具會監視網路中的策略點，以記錄並監視網路和應用程式的中斷情況。我們已建立了簡單的應用程式，可輕鬆又快速地找出網路中斷情形、應用程式故障、防火牆問題及 DNS 問題。此應用程式在邏輯方面似乎已臻成熟，還是才剛起步？

是否可以透過 Lotus Sametime 加強其主動性，而不是被動回應？到目前爲止，Healthcheck 是由一位操作人員來監視網路，然後只將問題傳給呼叫該工具的人員。這個方法對除錯及只有一個使用者固定監看主控台的環境很有效，但若要更臻完善，此應用程式必須發揮更多功能。

Lotus Sametime 之所以雀屏中選，是因爲其通知的速度和彈性卓越，勝過其他常見的通知方法（電子郵件、傳呼等）。電子郵件並非即時的通訊形式，因爲需要花數分鐘時間傳送故障通知（視傳訊基礎架構而定）。傳呼雖然是即時的，但需要管理者執行動作才能瞭解中斷的服務項目及其原因。Lotus Sametime 可讓 Healthcheck 提供即時通知一或多個使用者已知問題的相關資訊，以採取立即行動。

將 Lotus Sametime 通知功能加入 Healthcheck

我們使用 Lotus Sametime 7.5.1 Software Development Kit 輕鬆將 Healthcheck 轉型成 Lotus Sametime bot 程式。爲了使用 Lotus Sametime 通知功能，我們指定了 Lotus Sametime 使用者 ID 和密碼，以便 Healthcheck 用來傳送通知。Healthcheck 會使用此資訊登入 Lotus Sametime，然後傳送通知給輸入檔所指定的一或多個使用者。

若要確保 Lotus Sametime 服務中斷時持續發送 Lotus Sametime 通知，Lotus Sametime 環境最好是有多餘資源；建議使用 Lotus Sametime Chat 叢集功能和負載平衡器進行自動失效接手。

Lotus Sametime 通知程序可分爲五大步驟：

1. 建立並登入 Lotus Sametime 階段作業。啓動 Notification 物件後，首先必須實例化新的 STSession 物件來建立 Lotus Sametime 階段作業。此 STSession 物件是所有 Lotus Sametime 功能的關鍵，必須在登入之前建立。清單 4 說明連到 Lotus Sametime 伺服器並發出警示的執行緒執行方法。

清單 4. Healthcheck 執行方法（說明 STSession 的建立程序）

```
                    public void run() {
try {
   if (debug)
      healthchecker.Printnow("Notification thread is now starting");

                              // if stsession already exists, then unload and re-do
                              if (stsession != null)
                              {stsession.stop();
                              stsession.unloadSession();
                              }

                              stsession = new STSession("AlertSession");
                              stsession.loadSemanticComponents();
                              stsession.start();

                              //Console message if we are debugging
                              sysOut("Session created");
                              //Log into the st sametime server now
                              stLogin(st_server, st_user, st_pw);



                              while (healthchecker.iscomplete()==false)
                              {sleep(2000);}

                              healthchecker.Printnow("Notification thread is now complete");


                    }
                    catch (DuplicateObjectException doe) {
                              try {
                              //doe.printStackTrace();
                              // if dup, then unload and re-do just in case
                              if (stsession != null)
                              {stsession.stop();
                              stsession.unloadSession();
                              }
```

```
                        stsession = new STSession("AlertSession");
                        stsession.loadSemanticComponents();
                        stsession.start();
                        //Console message if we are debugging
                        sysOut("Session created");
                        stLogin(st_server, st_user, st_pw);
                        }
                        catch (DuplicateObjectException doe1)
                        {
                                //After unloading and trying again, just fail
                                sysOut("DuplicateObjectException thrown again");
                                healthchecker.setcomplete();
                                //destroy();
                        }


                }
                catch (Exception e)
                {
                        System.out.println("We failed getting the sametime session");
                }
        }
```

從此常式呼叫的 stLogin 方法會讓使用者登入 bot 程式。清單 5 說明簡單的登入常式。

清單 5. 簡單的登入常式

```
public void stLogin(String server, String user, String password)
        // Login routine to ST server.
        {
                commService =
```

```
                    (CommunityService) stsession.getCompApi(CommunityService.COMP_NAME);
                    commService.addLoginListener(this);
                    commService.loginByPassword(server, user, password);
                    sysOut("Attempting login to " + server);
          }
```

2. 順利登入後提供動作。順利完成 commService.loginByPassword() 方法後，應用程式必須提供由接聽器物件呼叫的 loggedIn() 方法，並傳送要通知的管理者名稱給此方法。請注意，這個類別要建立 InstantMessagingService 讓 Healthcheck 接聽這類活動，確定我們的物件是建立 Instant Message 時所呼叫的對象。

   此方法結束時，一旦 Services 建立完畢，就會呼叫 Resolver 服務查閱要通知的使用者名稱。清單 6 是順利登入 Lotus Sametime 後要執行的應用程式碼。服務一定要啓動，我們便嘗試解析接收警示的使用者。

清單 6. 說明服務啓動及解析器的程式碼

```
public void loggedIn(LoginEvent le)
          // Called after successful login
          // method will issue resolve of user to notify
          {
sysOut("Logged In");

//Attempt to set our status
try {
          STUserStatus status = new STUserStatus
          (STUserStatus.ST_USER_STATUS_DND, 0, STATUSMSG);
          le.getLogin().changeMyStatus(status);
}
          catch (Exception exc) {
          sysOut("Exception caught - " + exc.getMessage());
}
```

```
sysOut("Register to listen for incoming messages");
//Register to listen for incoming messages
//We aren't concerned about failures so no need for a try/catch
imService = (InstantMessagingService)
stsession.getCompApi(InstantMessagingService.COMP_NAME);
imService.registerImType(ImTypes.IM_TYPE_CHAT);
imService.addImServiceListener(this);

sysOut("Get a handle to the Lookup Service and add a resolve listener");
//Get a handle to the Lookup Service and add a resolve listener
lookupService = (LookupService) stsession.getCompApi(LookupService.COMP_NAME);
resolver = lookupService.createResolver(true, false, true, false);
resolver.addResolveListener(this);



// The way healthcheck class uses this class, only one name is supplied at once
//  May want to change in future to resolve all at once.



String [] copy = new String[name_count];
sysOut("Name count is:" + name_count);
           for ( int idx = 0; idx < name_count; idx++ ) {
           sysOut("copy complete;  copied name:" + users_to_notify[idx]);
                      copy[idx] = users_to_notify[idx];
                      }

sysOut("doing the resolve,  length is " + copy.length);
for (int i=0;i<copy.length;i++)
{
sysOut("names are " + copy[i]);
}
resolver.resolve(copy);




           }
```

3. 建立及傳送 IM 階段作業。現在已呼叫 resolver.resolve() 方法，是否要提供 resolved() 方法（成功呼叫 resolve() API 時就會呼叫此方法）則取決於 Healthcheck。以本案例來說，一旦在 Lotus Sametime 目錄中找到系統管理者的名稱，即可建立 Lotus Sametime IM 物件，然後傳送實際的應用程式或伺服器服務中斷清單。

   清單 7 說明成功解析 IM 目標使用者時執行的程式碼；一旦解析後，就會傳送實際的 IM。

清單 7. 建立 InstantMessagingService 及開啓 IM 階段作業的程式碼範例

```
public void resolved(ResolveEvent re) {
                    /*
                              if (re.getResolved() instanceof STUser) {
                              STUser user = ((STUser) re.getResolved());
                              String userName = user.getName();
                              sysOut("Resolved " + userName);
                              watchList.addItem(user);
                              sysOut("Added " + userName + " to WatchList");
                    */

                    sysOut("We resolved the name");
                    if (re.getResolved() instanceof STUser) {

                              // User was resolved.   Now get the IM service
                              STUser partner = ((STUser) re.getResolved());
                              imService= (InstantMessagingService)
                              stsession.getCompApi(InstantMessagingService.COMP_NAME);


                              String userName = partner.getName();
                              sysOut("Resolved " + userName);
                              //Set the IM Type to IM_TYPE_CHAT
                              int imt = ImTypes.IM_TYPE_CHAT;

                              //Set the encryption level
```

```
                        EncLevel enc = EncLevel.ENC_LEVEL_ALL;

                        //Create the IM so we can send the message
                        Im im = imService.createIm(partner, enc, imt);
                        im.addImListener(this);
                        // open will call imOpened in this class
                        im.open();
                    }

            }
```

請注意，resolved() 方法結束時，我們實際建立了 IM_TYPE_CHAT 類型的 IM 物件。這個物件會讓應用程式取得定義的 ImOpened() 物件，再用該物件傳送實際訊息。此案例傳送的訊息，是無法順利開啓 Socket 到主機或連接埠的清單。

一旦呼叫 Im.open() 方法，Healthcheck 就會提供 ImOpened() 啓動接聽器。實際上，傳送訊息到通知清單並關閉階段作業的就是這個常式。清單 8 說明 ImOpened 方法，成功呼叫 Im.open() 後就會啓動接聽器作業。

清單 8. 說明實際傳送文字訊息給管理者的 ImOpened 方法

```
public void imOpened(ImEvent ie)
           // method does the actual send of the IM once the IM was successfully opened
            {
sysOut("IM Opened to " + ie.getIm().getPartner().getName());
//To get the user status we need to cast the Im Object as an STWatchedUser

// if open, we are ready to send the IM
if (ie.getIm().isOpen()) {

           // Create message from all the failure reported in healthcheck class
           String text="HeathCheck reporting problems on the following ports:¥n";
           for (int loop=0;loop<error_index;loop++)
```

```
                                {
                        text=text.concat(Errors[loop] + "¥n");

                }

                //send the IM
                ie.getIm().sendText(false, text+"¥n");
                im_sent++;
                healthchecker.Printnow("Message sent to " +
                ie.getIm().getPartner().getName());



                try {
                        Thread.sleep(2000);
                }
                catch (InterruptedException e) {
                }

                // close the IM and start to tear everything down.
                ie.getIm().close(STError.ST_OK);
                healthchecker.setcomplete();

}
                }
```

順利透過 sendText 方法傳送訊息後，就會呼叫 IM 物件上的 close() 方法，然後再呼叫 destroy() 方法物件開始登出程序並分割 STSession，否則下次嘗試通知的新階段作業時，就會擲出 DuplicateSession 異常。

4. 關閉即時訊息。ImEvent 物件呼叫 close() 方法後，imClosed 方法就會移除處理即時訊息通道的接聽器。清單 9 說明簡單的 imClosed 常式。

清單 9. 簡單的 imClosed 常式

```
public void imClosed(ImEvent ie) {
                    sysOut("IM Closed from " + ie.getIm().getPartner().getName());
                    //The channel is closed so we can remove the listener
                    ie.getIm().removeImListener(this);
          }
```

5. 登出並結束 Lotus Sametime 階段作業。該階段作業關閉後，Healthcheck 就會登出 Lotus Sametime，同時結束步驟一所建立的 STSession 物件，teardown() 方法會被用來發出登出指令並結束該階段作業。清單 10 爲 teardown 方法和 LoggedOut 方法。

清單 10. 清除 Lotus Sametime bot 的 Teardown() 方法

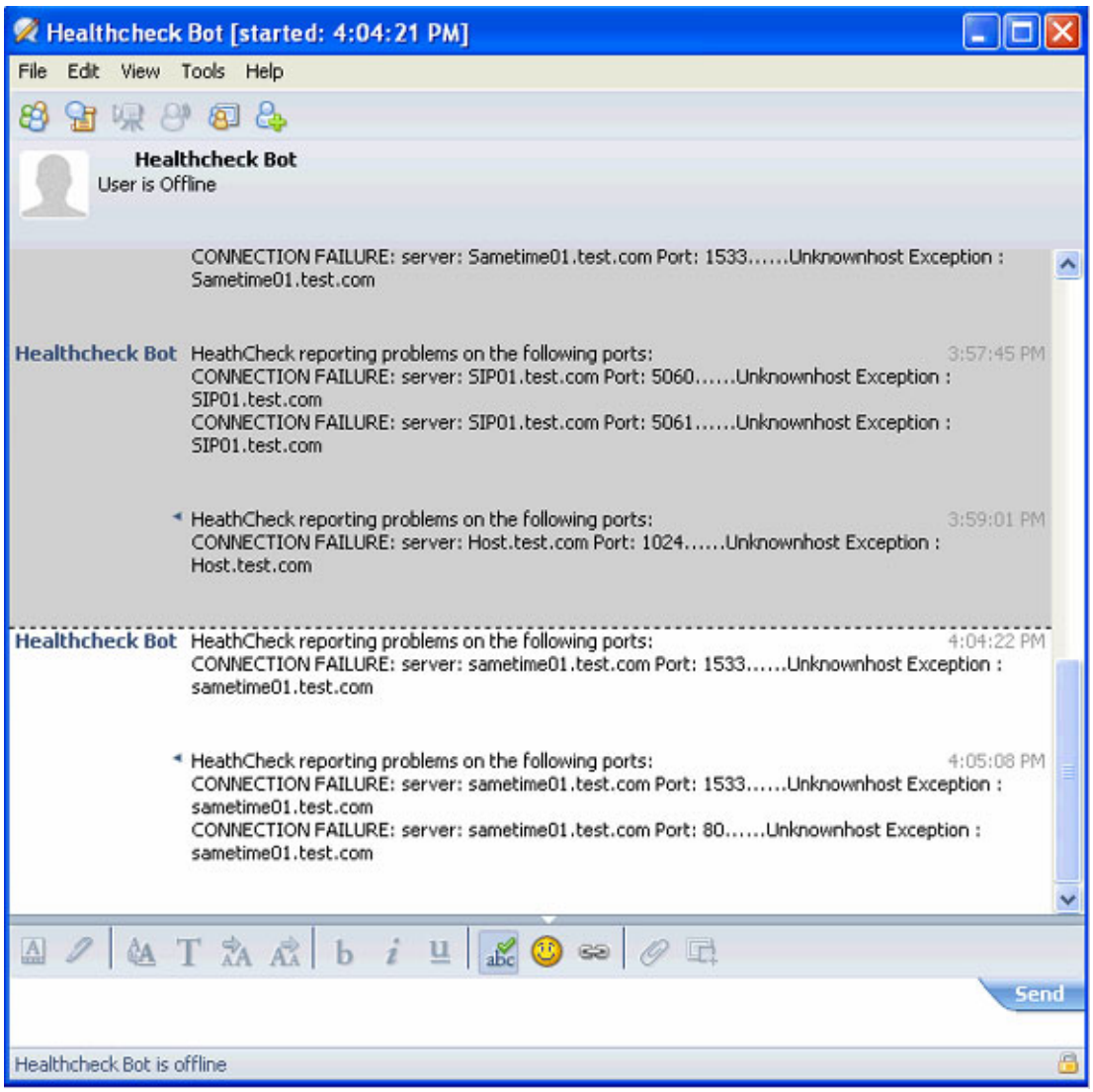
```
public static void teardown()
          // destroy is called whenever we are unloading the session
          after IM is sent or if failure occurs
           {
sysOut("Destroy called");
if (debug)
          healthchecker.Printnow("Notification thread is now complete");

if (commService != null && commService.isLoggedIn()) {
          try {
                    commService.logout();
          }
          catch (Exception exc) {
                    sysOut("Could not logout:" + exc.toString());
          }
}

try{
if (stsession != null)
          {stsession.stop();
```

```
            stsession.unloadSession();
            sysOut("Sametime Session unloaded");
            }
}

catch (Exception exc1) {
                        sysOut("Could not logout:Could not unload session object " +
                        exc1.toString());
            }

            }
```

6.

順利分割後，就會釋出用來產生通知的 Lotus Sametime 資源。最後就是要通知 Healthcheck，說明已經完成通知執行緒，然後繼續下一個通知或主機監視疊代。圖 3 說明 Healthcheck 發出的 Lotus Sametime 通知。

圖 3. Healthcheck 發出的 Lotus Sametime 通知



Healthcheck 輸出的通知清單內容包括：

- 服務中斷時間
- 伺服器或主機名稱
- 埠號
- 異常類型（Socket 開放呼叫擲出的三種異常之一）
- 傳回原因，位於 Exception 物件文字欄位

Healthcheck 每次通知都會登入及登出 Lotus Sametime。若 Healthcheck 在監視期間登入 Lotus Sametime，我們就需要內建重新連線邏輯，確保 Lotus Sametime server 服務中斷時 Healthcheck 可以自動進行失效接手。雖然此方法不是最快的傳送方式，但卻可以讓 Healthcheck 善用多餘的 Lotus Sametime 環境來傳送通知。

如果您使用的是較新版的 Lotus Sametime Eclipse 型用戶端（7.5 以及更新版本），也可以善用自動會談記載來保留服務中斷的連續記錄。您可從通知視窗建立群組會談，以納入瞭解問題詳情或如何解決問題的其他管理員。此外，可自訂 Healthcheck 加入其他回應資訊（雖然我們並沒有加入此功能）。

總結

本文說明了如何建立連接埠檢查程式搭配使用 Lotus Sametime，以提供服務中斷的即時通知。Lotus Sametime 工具集功能強大，可建立運用 Lotus Sametime 技術的自訂應用程式。希望本文對您有所啓發，讓您在 bot 程式中有效使用 Lotus Sametime 功能。

若對此資源有興趣且有意瞭解進一步詳情，請聯絡 IBM Software Services for Lotus (ISSL)。

資源

學習

- 閱讀 developerWorks® Lotus「A tour of the IBM Lotus Sametime V7.5 toolkits」一文。
- 閱讀 developerWorks Lotus「Creating a stock quote bot using the IBM Lotus Sametime Java toolkit」一文。
- 閱讀 developerWorks Lotus「Building a Lotus Sametime bot for language translation」一文。
- 閱讀 developerWorks Lotus「Building your own Sametime bots, Part 1」一文。
- 閱讀 developerWorks Lotus「Building your own Sametime bots, Part 2.」一文。
- 閱讀 developerWorks「developerWorks New to SOA and Web Services」網頁。

取得產品與技術

- 下載 Lotus Sametime SDK，其中包括 developerWorks Lotus 的 Lotus Sametime Java 工具集。
- 至 IBM Lotus Sametime 7.5.1 demonstration site 註冊。

討論

- 參加討論區。
- 閱讀 developerWorks Lotus Team blog。

作者簡介

Jim Dewan 是 IBM Lotus 和 Verozon 的資深支援經理，目前負責設計一系列工具/bot 程式，協助客戶監視及解除 Lotus 部署問題。Jim 有十年的 Lotus Domino 部署經驗，之前是 Lotus Domino Administration 團隊的專案負責人。此外，Jim 也是 Lotus Domino/Linux on Systemz 工作的技術主管，專精於應用程式

開發、工具集及企業資料協助工具。

Jennifer Wales 是經過 IBM 認證的 IT 諮詢專員，任職於 IBM 軟體事業處的 Workplace Portal and Collaboration (WPLC) 服務部門。她在網路整合業務方面已累積 19 年的 IT 專業經驗，負責範圍包括系統諮詢到專案管理。Jennifer Wales 善於運用 Lotus 技術設計複雜且高標準的多系統解決方案。她的專業領域包括 IBM Lotus Domino Server 架構及 IBM Lotus Sametime 即時傳訊。