



Cognitier SimpleIPC

Creating MS Word Doc from PHP Sample

Version 1.0.0.3

April 8, 2010

Copyright © 2010 Cognitier, Inc. All rights reserved.

Cognitier and the Cognitier logo are trademarks of Cognitier, Inc. All other company and product names referred to may be trademarks of their respective owners. This documentation is copyrighted material and is intended exclusively for use by licensed users of Cognitier software. The information in this document is subject to change without notice.

Creating MS Word Doc from PHP Sample

One of the least expensive and most widely-available web site hosting options is shared space on a Linux server, with the available web application coding option of PHP. There are many benefits to this configuration, but there are also a few drawbacks:

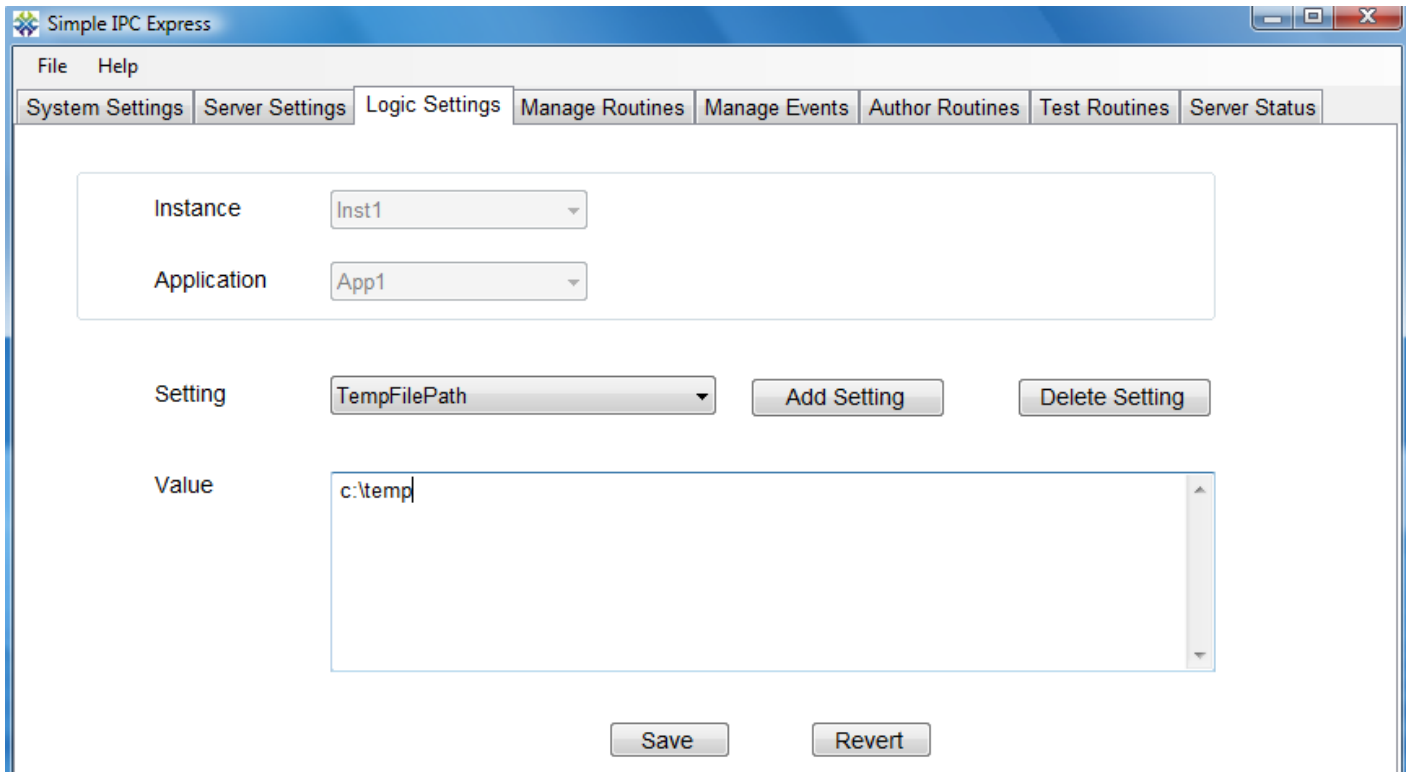
- Many web developers prefer to write code in .NET, and this coding option is not available on Linux.
- There may be a need to deliver the output of an application or service which only runs on the Microsoft Windows operating system. In this exercise, the application in question is Microsoft Word.
- There may be a need within the web application code to perform operations requiring elevated privileges on the machine, or to make system-wide settings changes, etc. This will not be allowed on a shared server.

The Messaging Peer For PHP product, in conjunction with SimpleIPC or SimpleIPC Express, allows the above drawbacks to be overcome by enabling the Linux-hosted PHP web application to delegate work to a .NET application running on a private (i.e. not exposed to the internet) Windows server. The following sample exercise assumes the reader has installed and configured either SimpleIPC or SimpleIPC Express, and also installed the Messaging Peer For PHP.

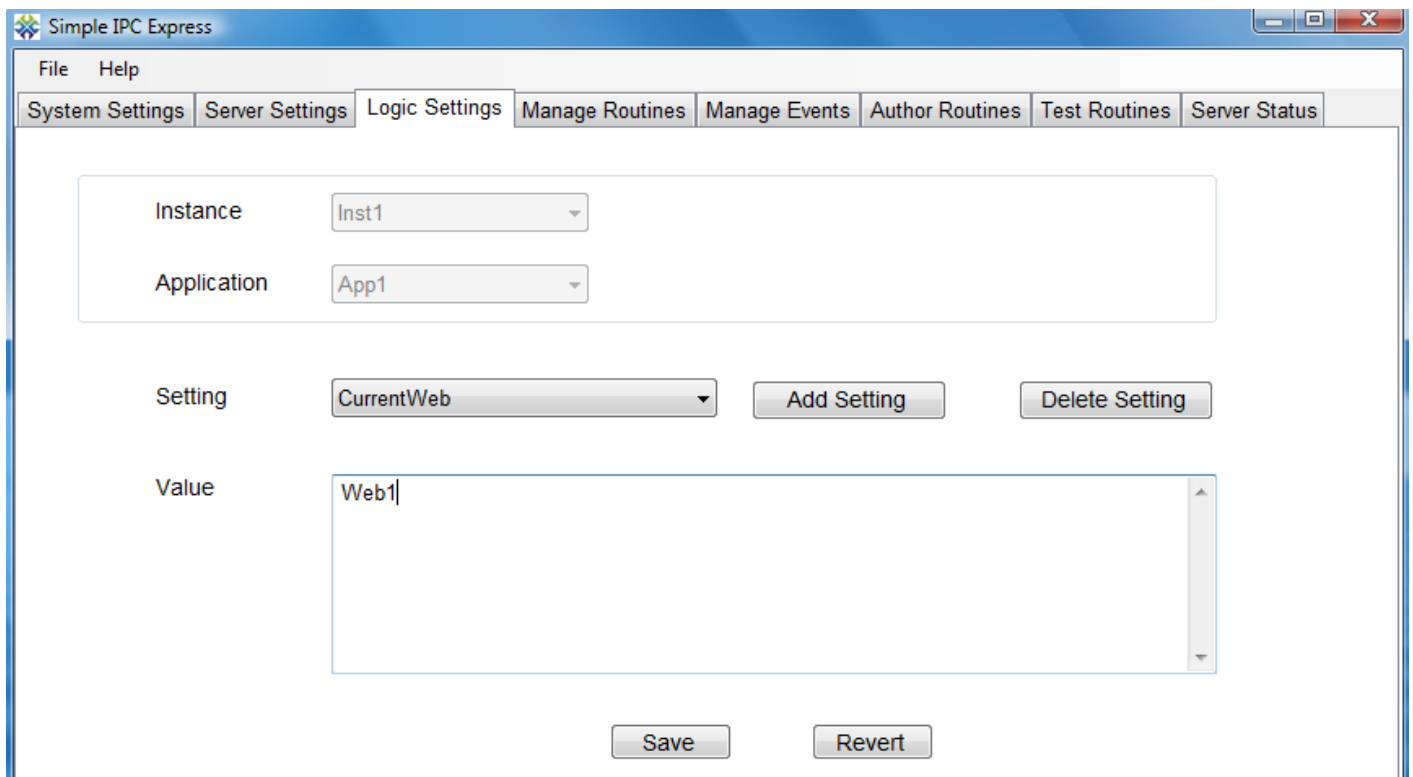
In this exercise, we will construct a guestbook application which creates a Microsoft Word document on the Windows server and returns that Word document to the web site visitor via the browser. When a web visitor enters his or her name on a web page hosted on the Linux server, a Word document with the entered visitor name will be rendered. This exercise will use instance “Inst1” in SimpleIPC on the Windows server. It assumes there is already an application called “App1”, and that there are no Events defined for this application. It assumes the name of the “web” that is configured in the Messaging Peer For PHP CTMsgWorkerConsole is “Web1”. Be aware of word-wrapping in the sample code listings.

This exercise assumes that Microsoft Word is installed on the Windows server. This exercise uses COM automation on the Windows server to construct the Word document. Depending upon the configuration on your Windows machine, you might experience errors which prevent the Word document from being constructed. Specifically, certain antivirus applications will interfere with the creation of the document via COM automation.

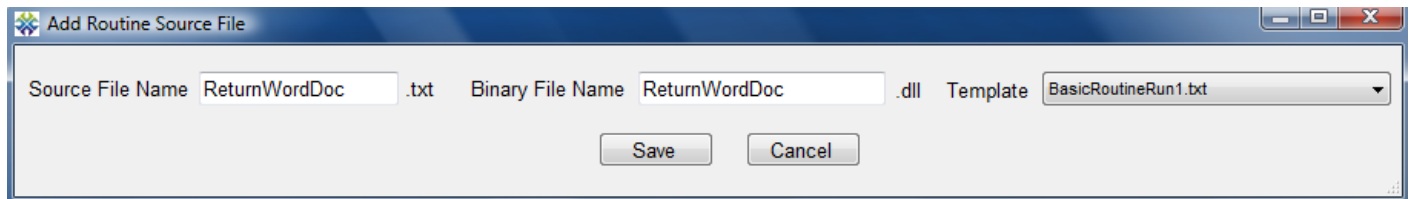
Step 1: Terminate any running IPC servers on the Windows server (the process name will be CTServerInst1.exe). On the Windows server, launch the SimpleIPC CTConsole. Go to the “Logic Settings” tab and create a setting for the name of the directory to which to write temporary files (if you do not already have a setting for this purpose). Name the setting “TempFilePath”, and give it a value appropriate for the machine (such as “c:\temp”). Save the setting.



Step 2: Create another Logic Setting with the name of the “web” configured in the CTMsgWorkerConsole. Name the Logic Setting “CurrentWeb”, and give it the value that matches the configured web name (“Web1”). Save the setting.



Step 3: Go to the “Author Routines” tab in SimpleIPC and create a new source file. Name the routine “ReturnWordDoc” and choose the BasicRoutineRun1 template.



Overwrite the code provided by the template with the following code:

```
import System;
import System.Collections;
import System.Reflection;

[assembly: AssemblyVersion("1.0.0.0")]

public class ReturnWordDoc implements CTSHost.IBasicRoutineRun1
{
    public function RunRoutine(oServerContextAccessor:
CTSHost.ServerContextAccessor, oSessionContextAccessor:
CTSHost.SessionContextAccessor, oCallParamsAccessor:
CTSHost.CallParamsAccessor) : System.Boolean
```

```

{
    var oLogUtil: CTCommon.LogUtil;
    var bRet: System.Boolean = false;

    oLogUtil = CTCommon.LogUtil.Instance;
    try
    {
        //Instantiate the MS Word COM Object and declare variables
        var oWordApp = new ActiveXObject("Word.Application");
        var oDoc = null;
        var oSelection = null;
        var sGuestName: System.String = String.Empty;
        var sFilePath: System.String = String.Empty;

        //Get the input argument with the web form field input - just one input in
this case
        if (oCallParamsAccessor.InputArgs.Count > 0)
        {
            sGuestName = oCallParamsAccessor.InputArgs[0];
        }
        //Create a temporary file name for the Word doc
        sFilePath = oServerContextAccessor.GetLogicSetting("TempFilePath") + "\\\"
+ oSessionContextAccessor.SessionID + ".doc";

        //Construct the Word document
        oDoc = oWordApp.Documents.Add();

        oSelection = oWordApp.Selection;
        oSelection.Font.Name = "Verdana";
        oSelection.Font.Size = "12";
        oDoc.Select();
        oSelection.TypeText("Dear ");
        oSelection.TypeText(sGuestName);
        oSelection.TypeText(",");
        oSelection.TypeParagraph();
        oSelection.TypeText("Thank you for using our application.");
        oSelection.TypeParagraph();
        oSelection.TypeText("Sincerely,");
        oSelection.TypeParagraph();
        oSelection.TypeText("The app support team");
        oDoc.SaveAs(sFilePath);
    }
}

```

```

oDoc.Close();

//Clean up objects
oSelection = null;
oDoc = null;
oWordApp.Application.Quit();

//Instantiate the client utilities object and upload the Word doc to the
server
var oCliUtil: CTMsgPeerPHPCliUtils.ClientUtility = new
CTMsgPeerPHPCliUtils.ClientUtility();
var oRetStruct: CTMsgPeerPHPCliUtils.UploadFileRetStruct =
oCliUtil.UploadFile(oServerContextAccessor.GetLogicSetting("CurrentWeb"),
sFilePath, "");

//Delete the local copy of the file
//System.IO.File.Delete(sFilePath);

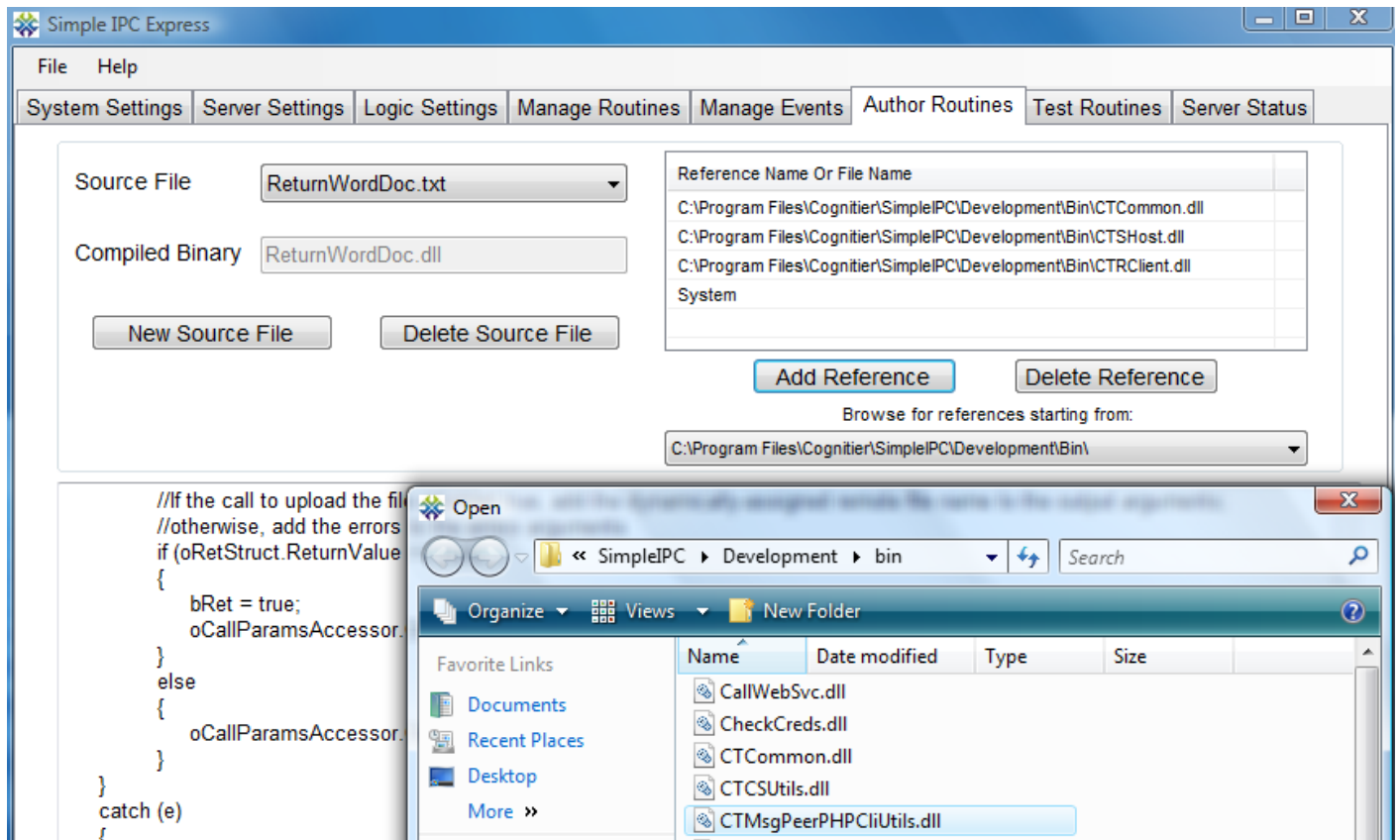
//If the call to upload the file returned true, add the dynamically-
assigned remote file name to the output arguments;
//otherwise, add the errors to the errors arguments
if (oRetStruct.ReturnValue == true)
{
    bRet = true;
    oCallParamsAccessor.OutputArgs.Add(oRetStruct.RemoteFileName);
}
else
{
    oCallParamsAccessor.OutputErrors.Add(oRetStruct.Errors);
}
}
catch (e)
{
    //Set errors into errors collection - must be strings
    oCallParamsAccessor.OutputErrors.Add(e.Message);

    //Also write errors to the server log file
    //Log levels are Errors (1), Warnings (2), Info (3), Verbose (4)
    oLogUtil.LogOutput(CTCommon.Constants.LogLevel.Errors,
CTCommon.Constants.LogOpType.RoutineExecution, e);
}
return bRet;

```

```
} //end function  
} //end class
```

Step 4: Add a reference to the “CTMsgPeerPHPCliUtils.dll” assembly. To do this, select the \Cognitier\SimpleIPC\Development\bin directory from the “Browse for references starting from” drop-down, and click the “Add Reference” button. Select the CTMsgPeerPHPCliUtils.dll file. After adding this reference, save and compile the new routine.



Step 5: Go to the “Manage Routines” tab in SimpleIPC in order to register the new routine. Click the “Refresh File List” button under the “File (assembly)” dropdown. Select the “ReturnWordDoc.dll” file, and enter a Routine Name of “ReturnWordDoc”. Save the new routine registration.

Register New Routine

File (assembly)

ReturnWordDoc.dll

Refresh File List

(C:\Program Files\Cognitier\SimpleIPC\Development\Bin)

Type (class name)

ReturnWordDoc

Interface

IBasicRoutineRun1

Routine Name

ReturnWordDoc

Save New Routine Registration

Step 6: Create a PHP file to be executed on the Linux server. The PHP file will call the ReturnWordDoc routine we just created and registered. Use an editor such as Notepad to create a PHP file called "Guestbook2.php" with the following code:

```
<?php
    include("CTUtils.php");
    $sLogDir = "log";

    $InstName = "Inst1";
    $AppName = "App1";
    $RoutineName = "ReturnWordDoc";
    $bStateless = true;
    $sSessionID = "";
    $sRoutineErrMsg = "";
    $sReturnedFileName = "";

    try
    {
        $ini_array = parse_ini_file("CTMPConfig.php");
        $sLogDir = $ini_array["logdir"];
        $sUploadDir = $ini_array["uploaddir"];

        $sReqID = "";
        $iTimeLimit = 120; //2 minutes
```

```

        //Create the base request document with the app, instance, routine,
etc.
        $oDoc = startReqDoc($InstName, $AppName, $RoutineName, $iTimeLimit,
$bStateless, $sSessionID, $sReqID);

        //Add routine input arguments to the request document based on form
inputs - make the call once per input.
        //If we wanted to have the messaging peer challenge credentials, we
would add the credentials as inputs here.
        addInArgToReqDoc($oDoc, $_POST["contactname"]);

        //Save the request document as a means of submitting the request to
the messaging peer
        saveReqDoc($oDoc, $sReqID);

        $oOutargs = array();
        $oErrors = array();
        $oWarnings = array();
        //Call getRespVals() to retrieve the response from the messaging
peer - this call is synchronous and may take some time
        if (getRespVals($sReqID, $oOutargs, $oErrors, $oWarnings,
$sSessionID, $iTimeLimit) == true)
        {
            if (count($oOutargs) > 0)
            {
                //The only return value we're expecting is the name of
the file that was built on the Windows server and uploaded.
                //We'll pick up this file from the local server and send
it back to the web client.
                $sReturnedFileName = $oOutargs[0];
            }
            if (strlen($sReturnedFileName) > 0)
            {
                header("Content-Description: File Transfer");
                header("Content-Type: application/octet-stream");
                header("Content-Disposition: attachment;
filename=Mydoc.doc");
                header("Content-Transfer-Encoding: binary");
                header("Expires: 0");
                header("Cache-Control: must-revalidate, post-check=0,
pre-check=0");
                header("Pragma: public");
            }
        }
    }
}

```

```

        header("Content-Length: " . filesize($sUploadDir . "/" .
$sReturnedFileName));
        readfile($sUploadDir . "/" . $sReturnedFileName);
        flush();
    }
    else
    {
        echo "<HTML><BODY><HEAD><TITLE>Post
Result</TITLE></HEAD>The server was unable to process your request at this
time. Please try again later.</BODY></HTML>";
    }
}
else
{
    //If the call to getRespVals() returns false, then loop through
the returned errors to build a single error message string.
    foreach($oErrors as $oError)
    {
        $sRoutineErrMsg = $sRoutineErrMsg . $oError . "<br>";
    }
    echo "<HTML><BODY><HEAD><TITLE>Post Result</TITLE></HEAD>The
server was unable to process your request at this time. Please try again
later.\n" . $sRoutineErrMsg . "</BODY></HTML>";
}

}
catch (Exception $eRequest)
{
    logErr($sLogDir . "/" . getMills() . "_" . getmypid() .
"_RecCont.log", $eRequest->getMessage());
    echo "<HTML><BODY><HEAD><TITLE>Post Result</TITLE></HEAD>The server
was unable to process your request at this time. Please try again later.\n" .
$eRequest->getMessage() . "</BODY></HTML>";
}
?>

```

Save the file and upload it to the Linux server. Place the file in the directory corresponding to your BaseURL setting.

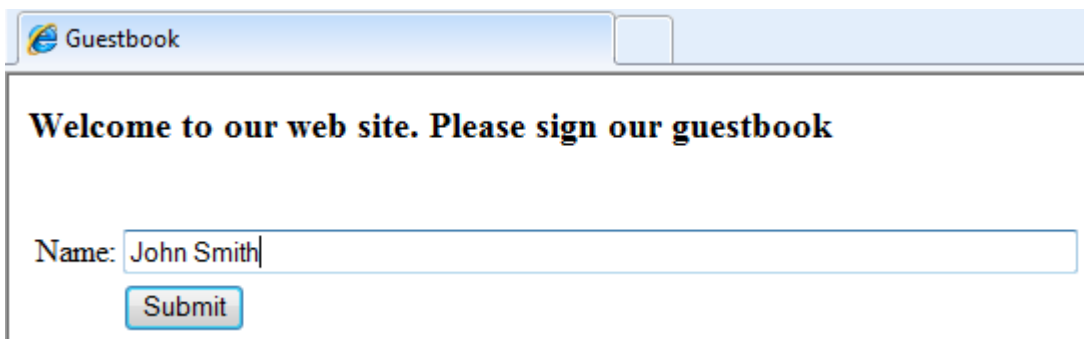
Step 7: Create a static HTML file to be hosted on the Linux server which will invoke the PHP file just created. Name the file "Guestbook2.htm". Use an editor such as Notepad to enter the following HTML for the web page:

```
<HTML>
```

```
<BODY>
<HEAD>
<TITLE>Guestbook</TITLE>
</HEAD>
<H3>Welcome to our web site. Please sign our guestbook</H3>
<BR>
<FORM ACTION="Guestbook2.php" METHOD="POST">
<TABLE>
<TBODY>
<TR>
<TD>Name:</TD><TD><INPUT TYPE="TEXT" NAME="contactname" SIZE="75" /></TD>
</TR>
<TR>
<TD></TD><TD><INPUT TYPE="SUBMIT" VALUE="Submit" /></TD>
</TR>
</TBODY>
</TABLE>
</FORM>
</BODY>
</HTML>
```

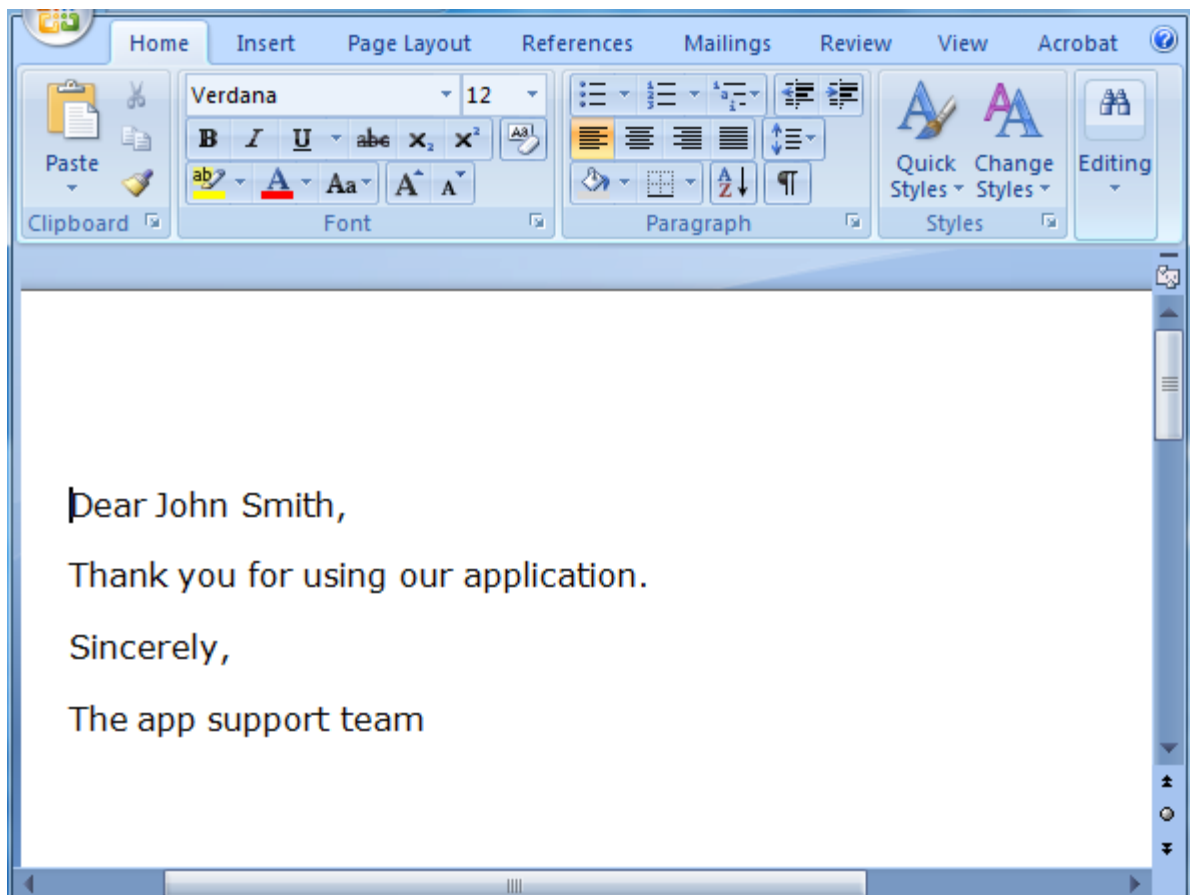
Save the file and upload it to the Linux server. Place the file in the directory corresponding to your BaseURL setting.

Step 8: Start the Messaging Peer service on the Windows server if it is not already running. Open a web browser and navigate to the HTML file we just created and placed on the Linux server. If the “BaseURL” setting for your “web” in the CTMsgWorkerConsole is [http://www.\[your public domain name\].com](http://www.[your public domain name].com), then navigate to [http://www.\[your public domain name\].com/Guestbook2.htm](http://www.[your public domain name].com/Guestbook2.htm). Enter a visitor name.



The screenshot shows a web browser window with the title "Guestbook". The main content of the page is a heading that reads "Welcome to our web site. Please sign our guestbook". Below the heading is a form with a label "Name:" followed by a text input field containing the text "John Smith". Below the input field is a button labeled "Submit".

Click the “Submit” button and ensure that a Word document is rendered with the expected contents.



The sample exercises demonstrated the exchange of requests and responses between the Linux server and the Windows server. Any tasks more appropriately performed by the Windows server can thus be delegated. Firewall issues can be bypassed such that the requests and responses can still be exchanged even if the Windows server is completely inaccessible from the public internet.