Integrity Document Library

Client Support Utility Guide

Using the Client Log Upload Utility

1-0280-0650-2005-10-05

Contents

Using the Client Log Upload Utility

This document describes Integrity client's Client Log Upload Utility.

Introducing the Client Log Upload Utility

Specially branded versions of Integrity Agent and Integrity Flex (referred to jointly as Integrity client in this document) include a Client Log Upload Utility.

The Client Log Upload Utility provides a way for a user to assist technical support personnel by uploading Integrity client diagnostic information to a pre-defined location. Technical support personnel can then copy uploaded files from the destination server and analyze the uploaded log data to help isolate and correct problems on the user's computer.

This document contains four sections:

- "Preparing a destination server," in the following section
- "Installing the Client Log Upload Utility," on page 2
- "Configuring the Client Log Upload Utility," on page 2
- "Using the Client Log Upload Utility," on page 4

Use the information contained in these four sections to prepare for, configure, and use the Client Log Upload Utility.

Preparing a destination server

The Client Log Upload Utility uses an `HTTP PUT` statement to upload Integrity client log files to the destination server; after the files have been received by the destination server, the files can then be forwarded to a location where technical support personnel then access the log data.

The following illustrates the general format of the HTTP Put call used by the Client Log Upload Utility:

```
http://lockup.company.com/error?filename=xyz
```

In the preceding example:

- `http://lockup.company.com/error` corresponds to the Client Log Upload Utility **File Destination** text entry area
- `xyz` corresponds to the Client Log Upload Utility **File Name** text entry area

See the procedure under "**Task 2: Editing an installation configuration file**," for an illustration of the two text entry areas.

See Appendix B, "Apache Web Server Scripts," for additional information about using a Perl script to configure an Apache Web Server to receive uploaded client debug logs.

Copying server address information to a configuration file

You must also supply the destination server address to the Client Log Upload Utility. See "Installing the Client Log Upload Utility," on page 2, for information about putting the server address into an installation configuration file.

Installing the Client Log Upload Utility

The installation program for specially branded versions of Integrity Agent and Integrity Flex automatically installs the following three files in the Integrity client installation folder.

- clientDiagUpld
- tvdebug.reg
- tvdebugoff.reg

See Appendix A, "Diagnostic Upload Utility Files," for a description of the files used by the Client Log Upload Utility.

Configuring the Client Log Upload Utility

After you have identified the destination server address, you are ready to put the address into an installation configuration file.

To enable the Client Log Upload Utility to upload client log information, you must:

- Enter the destination server address and folder information into an installation configuration file
- Create an installation command line that directs Integrity client to read the installation configuration file at the time of installation

This section includes three tasks:

- "Task 1: Obtaining a configuration file," in the following section
- "Task 2: Editing an installation configuration file," on page 3
- "Task 3: Creating an installation command line," on page 4

Perform each of these tasks in the order listed to specify the location of the log upload server and folder, and to have Integrity client read location of the server at the time of installation.

Task 1: Obtaining a configuration file

The Client Log Upload Utility reads the destination server address and folder name from an installation configuration file. Complete the following procedure to save Integrity client settings in an XML configuration file. You can then use the procedure under "Task 2: Editing an installation configuration file," on page 3, to enter the destination server address into the file.

To save Integrity client settings as an XML configuration file:

1. In the Windows System Tray, double-click the Integrity icon to open the Integrity client program's Control Center.

   Depending on the current operating mode of the Integrity client program, the Integrity Agent or Integrity Flex Control Center appears.

2. In either Integrity Agent or Integrity Flex:

   a. Select the **Policies** panel.

      The Policies panel appears. There may be two policies listed, depending on whether or not Integrity Flex or Integrity Agent is connected to an Integrity Server.

   b. In the **Policies** panel, select the **Personal** policy from the list of policies then hold down CTRL and ALT and double-click.

   c. The computer's text editing program appears. (Windows' default text editor is Notepad). The text editing program contains the current settings for Integrity Flex or Integrity Agent in XML format.

3. In the text editing window use the **Save As** feature to save the policy file to the `C:\Program Files\Zone Labs\Integrity Client` folder as *FileName*`.xml`. The Integrity client program accepts any valid Windows file name, but the `.xml` file name extension must be used.

Proceed to "Task 2: Editing an installation configuration file," in the following section to specify the value of `ErrorLogUploadDirectory` in the configuration file.

Task 2: Editing an installation configuration file

Use the `configuration` element's `ErrorLogUploadDirectory` attribute to provide the Client Log Upload Utility with the destination server address and folder to receive uploaded logs.

The following illustrates the general form of the `configuration` element of an XML Policy file as well as the placement of the `ErrorLogUploadDirectory` attribute.

```
<ZoneLabsSettings … />
  <ruleset name="runningruleset" start="afterstartup" stop="onshutdown">
    <configuration … ErrorLogUploadDirectory="http://lockup.zonelabs.com/error" …
    />
  </ruleset>
```

You must maintain the configuration file's hierarchical relationship of XML elements when editing the `ErrorLogUploadDirectory` attribute to an XML Policy file.

Complete the following procedure to specify the `configuration` section's `ErrorLogUploadDirectory` attribute.

To edit an installation configuration file:

1. Open the XML configuration file in a text editor. The default text editor for Windows is Notepad, although you may prefer to use an editor designed specifically to work with XML files.
2. Locate the `configuration` element in the XML Policy file:
   a. In the text editor, search or browse for the `ruleset` element that contains the `name="runningruleset"` attribute.
   b. From the line containing the `name="runningruleset"` attribute, search or browse forward to the `configuration` element. The `configuration` element is several lines below the running ruleset element definition.
3. Specify the desired operating mode. Inside the `configuration` element, type the following, including the quotation marks (line breaks added for readability):

   ```
   <configuration …
   ErrorLogUploadDirectory="http://serverName.com/folderName" …
   />
   ```

   In the preceding example, `ServerName/folder` name is the destination server address and folder for uploaded client error logs.
4. Save the change to the XML Policy file and close the text editing tool.

Proceed to "Task 3: Creating an installation command line," on page 4, to create a batch file containing an operational command line.

Default value of ErrorLogUploadDirectory

If neither the `ErrorLogUploadDirectory` nor the `LockupRedirect` and `server` attributes have been specified, the value of `ErrorLogUploadDirectory` defaults to:

```
http://lockup.zonelabs.com/upload
```

Task 3: Creating an installation command line

The final task required to change operational modes is to create an Integrity client installation command line. The command line forces the Integrity client program to read a new value for `ErrorLogUploadDirectory` from an installation configuration file.

Refer to the Client Installation Guide for more details about installation command lines.

Using the Client Log Upload Utility

The Client Log Upload Utility does not run automatically: No Integrity client error condition initiates automatic upload of diagnostic information. Instead the user responds to a request from a technical support representative by running Client Log Upload Utility.

The following procedure describes how to run the Client Log Upload Utility.

To run the Client Log Upload Utility:

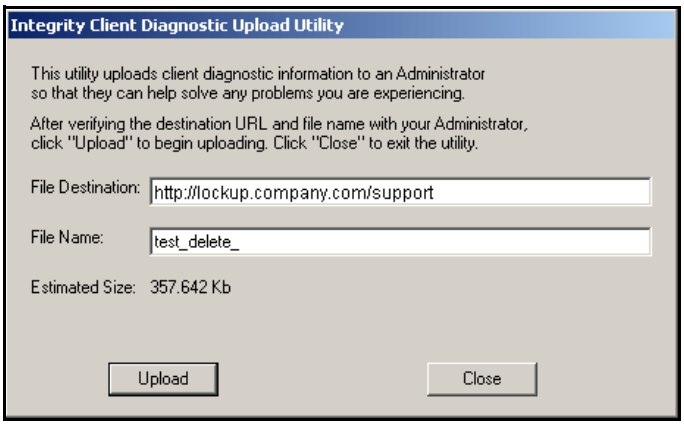
1. Browse to the folder that contains the Integrity client program.

   The default location for the Integrity client program is
   `C:\Program Files\Zone Labs\Integrity Client\`

The CLUU Icon

2. In the Integrity Client folder, double-click the Client Log Upload Utility (CLUU) icon (shown at left).

   The Client Log Upload Utility displays the file destination and name dialog box.



The file destination and name dialog box

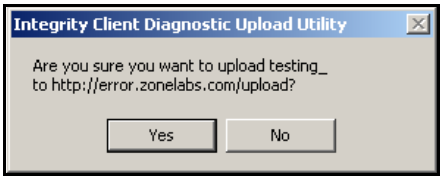
   The Client Log Upload Utility:

   - Automatically copies the destination server address into the **File Destination** text entry area
   - Automatically creates a unique file name by appending a date and time stamp to the network domain and username of the endpoint computer in the **File Name** text entry area

   Technical support personnel can also direct the user to modify destination upload server address and folder, and file name information at the time the Client Log Upload Utility is run.

3. In the file destination and name dialog box, click **Upload**.

   The Client Log Upload Utility displays an upload confirmation dialog box.



The upload confirmation dialog box

4. In the **upload confirmation** dialog box, click **Yes**.

   The Client Log Upload Utility:

- Begins uploading the diagnostic information to the destination server
- Updates the file destination and name dialog box with the progress of the upload
- Displays a message in the file destination and name dialog box when the upload has completed successfully

5. After the upload has completed, click **Close** to exit the Client Log Upload Utility.

Updating the destination server address

As described under "Installing the Client Log Upload Utility," on page 2, the Client Log Upload Utility normally receives the destination server address from an installation configuration file.

If Integrity client receives an updated destination server address in an enterprise security policy, it uses the newer address. If there is more than one enterprise policy present on the computer, the Client Log Upload Utility uses the active or most recently enforced version policy.

Appendix A

Diagnostic Upload Utility Files

Table A-1 describes the files used by the Diagnostic Upload Utility.

Filename	Purpose	Value
tvdebug.reg	Changes Integrity client's `TVDebug Log` flag to -1 When the `TVDebug Log` flag equals `-1`, Integrity client captures and stores a large amount of additional debugging information. This setting should only be applied on a temporary basis for purposes of troubleshooting.	xFFFFFFFF
tvdebugoff.reg	Changes Integrity client's TVDebug Log flag to 1 When the `TVDebug Log` flag equals `1`, Integrity client captures and stores only basic event log information. This setting should applied for normal operation.	x00000001
clientDiagUpld	Shortcut for starting the Diagnostic Upload Utility program file. The shortcut contains the following command line (line breaks added for readability): "C:\Program Files\Zone Labs\Integrity Client\ zlclient.exe" -errorsubmission -integrityInteractive The `errorssubmission` and `integrityInteractive` operational command line switches used to run the Diagnostic Upload Utility function only with specially branded versions of Integrity Agent and Integrity Flex.	

Table A-1: Files used by the Diagnostic Upload Utility

Appendix B

Apache Web Server Scripts

This appendix provides an example of a Perl script that configures an Apache destination server to receive uploaded log information from the Client Log Upload Utility. This appendix contains two sections.

- "Creating an Apache Web Server Perl script," in the following section
- "Sample put.pl file," on page 9

Use the samples in these sections as a guide to creating a Perl script to configure an Apache Web Server for use with the Client Log Upload Utility.

Creating an Apache Web Server Perl script

You can use a Perl script to automate the configuration of a destination server for use with the Client Log Upload Utility. The following procedure illustrates one way to add Client Log Upload Utility information to an Apache server's `httpd.conf` file.

Before you begin

The following example assumes that DocumentRoot equals:

```
C:/Program Files/Apache Group/Apache2/htdocs
```

To create a Perl script:

1. In the destination Apache Web server, add the following entry to the `httpd.conf` file found in `<apache_home>/conf` directory:

   ```
   <Directory "C:/Program Files/Apache Group/Apache2/htdocs/upload">
      Script PUT /cgi-bin/put.pl
   </Directory>
   ```

2. Save the `httpd.conf` file

   The PUT command now points to the Perl script file, `put.pl`, described under "Sample put.pl file," in the following section.

3. Place the `put.pl` file in the `<apache_home>/cgi-bin` directory.
4. Stop, then restart the Apache Web server.

Sample put.pl file

The `put.pl` file, referred to under "Creating an Apache Web Server Perl script," in the preceding section, provides a simple HTTP PUT command handler. The following code sample illustrates the general form of a `put.pl` file.

```
#!/usr/local/bin/perl
# Very simple PUT handler. Read the Apache Week article before attempting
# to use this script. You are responsible for ensure that this script is
# used securely.

# A simple log file, must be writable by the user that this program runs as.
# Should not be within the document tree.
$putlog = "put1.log";
open (DEBUG, ">>c:\\somefile.out") || die "unable to open output file: $!\n";

# Check we are using PUT method
print DEBUG ("request method is $ENV{'REQUEST_METHOD'}");
if ($ENV{'REQUEST_METHOD'} ne "PUT") { &reply(500, "Request method is not PUT"); }

# Note: should also check we are an authentication user by checking
# REMOTE_USER

# Check we got a destination filename
$filename = $ENV{'PATH_TRANSLATED'};
print DEBUG ("filename is $filename");
if (!$filename) { &reply(500, "No PATH_TRANSLATED"); }

# Check we got some content
$clength = $ENV{'CONTENT_LENGTH'};
print DEBUG ("content length is $clength");
if (!$clength) { &reply(500, "Content-Length missing or zero ($clength)"); }

# Read the content itself
$toread = $clength;
$content = "";
while ($toread > 0)
{
   $nread = read(STDIN, $data, $clength);
   &reply(500, "Error reading content") if !defined($nread);
   $toread -= $nread;
   $content = $data;
}

# Write it out
# Note: doesn't check the location of the file, whether it already
# exists, whether it is a special file, directory or link. Does not
# set the access permissions. Does not handle subdirectories that
# need creating.
open(OUT, "> $filename") || &reply(500, "Cannot write to $filename");
print OUT $content;
```

```
close(OUT);
print DEBUG ("reached here ...");

# Everything seemed to work, reply with 204 (or 200). Should reply with 201
# if content was created, not updated.
&reply(201);

exit(0);

#
# Send back reply to client for a given status.
#

sub reply
{
   local($status, $message) = @_;
   local($remuser, $remhost, $logline) = ();

   print "Status: $status\n";
   print "Content-Type: text/html\n\n";

   if ($status == 200) {
      print "<HEAD><TITLE>OK</TITLE></HEAD><H1>Content Accepted</H1>\n";
   } elsif ($status == 500) {
      print "<HEAD><TITLE>Error</TITLE></HEAD><H1>Error Publishing File</H1>\n";
  print "An error occurred publishing this file ($message).\n";
   }
   # Note: status 204 and 201 gives have content part

   # Create a simple log
   $remuser = $ENV{'REMOTE_USER'} || "-";
   $remhost = $ENV{'REMOTE_HOST'} || $ENV{'REMOTE_ADDR'} || "-";

   $logline = "$remhost $remuser $filename status $status";
   $logline .= " ($message)" if ($status == 500);
   &log($logline);
   exit(0);
}

sub log
{
   local($msg) = @_;
   open (LOG, ">> $putlog") || return;
   print LOG "$msg\n";
   close(LOG);
}
```