Technote 37 - Obvius Upload Kit Overview

Obvius
Mike Hewitt
08/16/10

Overview

Using its integrated modem or Ethernet (LAN) connection, the AcquiSuite can push or pull data via HTTP, XML, FTP or any custom protocol utilizing the AcquiSuite Module to build your own application. Obvius offers several solutions for gathering data log files from AcquiSuites or AcquiLites, including the Building Management Online service for uploads, and EnertraxDL. Additionally, many Obvius software partners offer full implementations of these solutions.

Purpose

For customers wishing to create unique hosted solutions on customer-based servers, Obvius offers an upload kit. The upload kit is a collection of documents and initial code pieces -- an example cookbook/toolbox for developers and systems administrators to demonstrate the communication methods that Obvius Data Acquisition Servers (DAS) use to converse with a host server. For example, a PHP programmer can localize some of the code and quickly establish an http connection to a web server process which can then communicate with an AcquiSuite using PHP responses to the AcquiSuite http packets. Some rudimentary error handling exists in the upload kit, which will typically report an error soon after log files have successfully arrived on a server and the unmodified server script refuses to accept the same file again as the DAS attempts to resend it. In a similar way, the example scripts contain little in regard to serial number or user authentication, which would be typical of a final fully-implemented robust solution.

The Next Step

Following the initial test to verify that the communication occurs, it is time to decide what functionality the server needs to interpret the incoming file, or store it, or delete, and so on. These decisions and the code to implement them are beyond the scope of the example upload kit. The remainder of this document is intended to help describe the general conversation between the DAS and the server.

Configuration

In order to make use of the upload kit example scripts, the DAS needs to direct uploads to a running server which is ready to recognize the http requests. Use the Setup / Upload page within Log File Data to Enable an Upload Channel to refer to that server. For example, Upload URL could contain a path directing the DAS to a RAS running a PHP script, such as

"http://www.customer_server_host/test_code/upload.php?"

Refer to the DAS manual for details about the Log File Data pages and DAS configuration. Also verify that the host server is able to support the script, such as file permissions, language extensions, and host configuration options, such as required php modules in an httpd.conf file.

Operation

When the DAS transmits information, such as outbound http packets, it will declare a "MODE". The example upload kit scripts demonstrate various possible server responses to the MODE condition declared by the DAS, such as a MODE of "STATUS". The responses provided by the sample upload kit scripts are only limited examples. Many more responses are possible.

As each packet arrives at the server, the value of MODE helps determine the best course of action. The flow in the following diagram shows possible conversations between the DAS and the server.

Note that the supplied example scripts cannot do any checking of user, password or DAS Serial Number, since these are specific to each site. Additional code work is needed on the server to provide serial number checking decision shown in the flow diagram.

If any server-specific issues such as language support are resolved, when the DAS sends a MODE of LOGFILEUPLOAD, it will also result in log files from the DAS arriving on the server. Soon after that, the next time that same log file arrives on the server, the example script storage operation will fail since that log file already exists on the server. The fully-implemented server solution would very likely have more complex log actions than the simple storage technique shown in the example upload kit scripts, such as using the configuration manifests to compare which log files exist on the server versus the client. Server evaluations of the MODE messages from the DAS for CONFIGFILEMANIFEST, CONFIGFILEDOWNLOAD and CONFIGFILEUPLOAD within the simple example upload kit scripts will likely function to some extent, but a production solution would far exceed those simple tests.

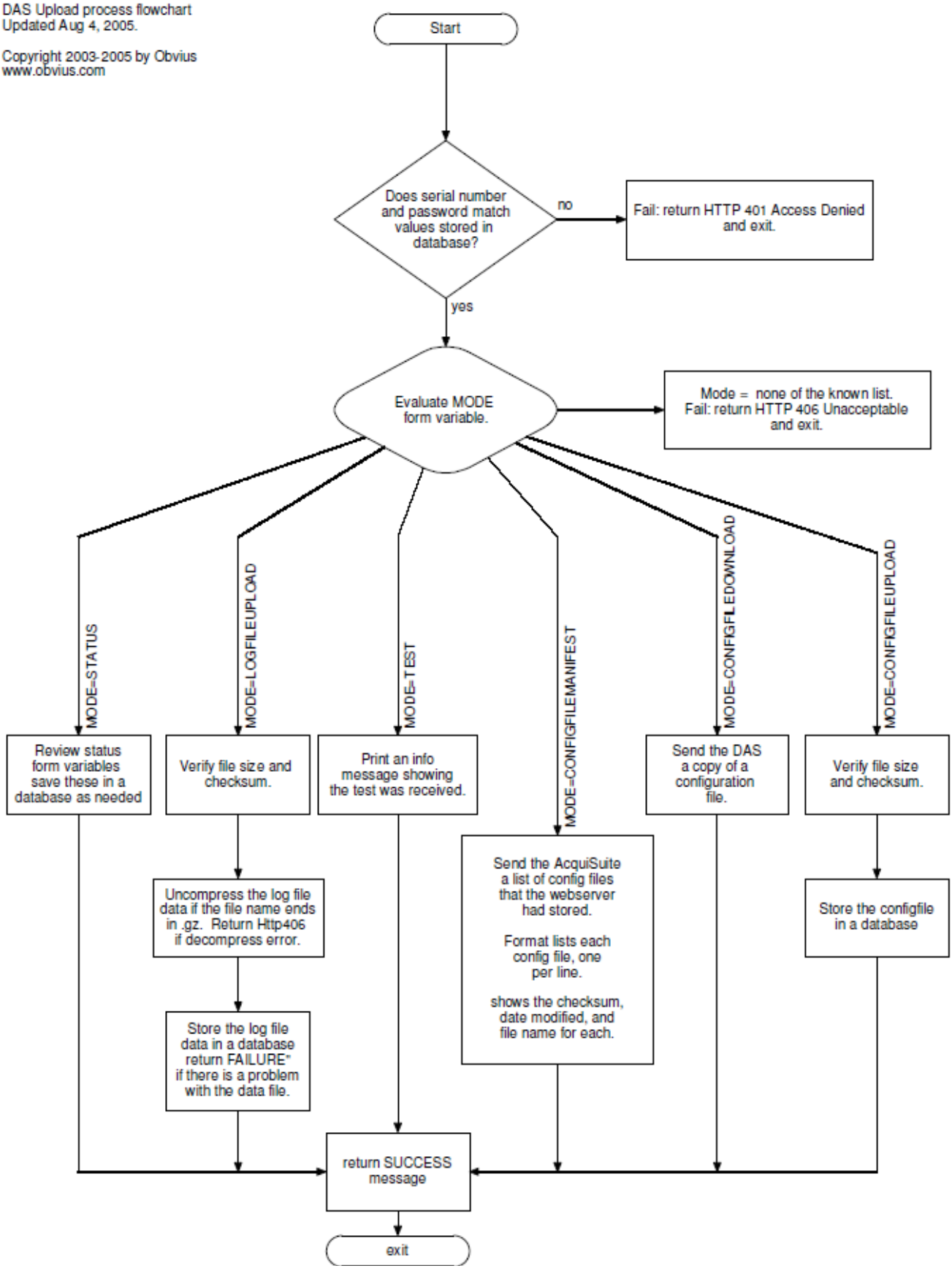
Note that the flow diagram is stateless. This means that each MODE condition is independent, so full use of the conversation with the DAS implies that the final multi-threaded server code would likely include state-sensitivity to prevent issues such as repeated uploads of the same file.

On the client side, the DAS is sensitive to the certain key words at the beginning of lines within the message bodies from the server. Characters in these words must be all upper-case:

- NOTES *do not cause any behavior change on the DAS, but will appear for the user when viewing upload connection test logs.*
- SUCCESS *signals to the DAS that the log file uploaded correctly*
- FAILURE *error conditions prevented the DAS or server from completing an action*

If the DAS does not receive the "SUCCESS" confirmation, it will repeatedly try to re-upload the same file on the next automatic or manual upload operation. In the same way, if the configuration file on the DAS does not match the configuration file on the server, the DAS will attempt uploads until it receives a matching configuration file. This sounds redundant, and it is, but for good reason. If the file uploads but gets garbled or is in some way not usable by the server (for example a dropped packet or line noise causes the checksum on the received copy to differ from the original checksum), then the file could be considered of no value, so it should be discarded and retried. This strategy allows the server to assure that the log file it processes is the same as the one recorded on the DAS, regardless of communication difficulties.

DAS Upload process flowchart
Updated Aug 4, 2005.



Diagnostics and Troubleshooting

Several helpful resources exist for the process of examining the DAS-RAS conversation, to aid in initial understanding of the example scripts in the upload kit as well as later analysis of the communication with the DAS from actual server scripts created on the customer host servers.

On the server side, the host server may be configured to output information or error messages into the browser or into various log files on the server. For example, an web server providing httpd service may report important messages in

/var/log/httpd/error_log

such as

PHP Notice: Undefined index NOT_STATUS in ...upload_test.php on line 103

which would indicate that the DAS sent the server a collection of named values, but none of them were named "NOT_STATUS" so whatever function on line 103 of the php script requested it failed to fill in the value. Again, this is just an example of the type of diagnostic information available in the server log.

On the client side, the DAS provides valuable tracing techniques. When accessing the DAS from a web browser, the level system logging detail can be configured as desired, including full debug logging. Off by default and usually off in typical operation, debug logging during file uploads records outbound and inbound traffic on the DAS, and can show all messages. Ordinarily only basic failure and success notices appear in the log. Since different DAS web interfaces may vary between products, be sure to look for debugging options on Log File Data pages and System pages in the DAS web menus. Also try trace routing the Connection Test.

The upload kit may contain short text notes about a particular file or topic, and much of the code includes contextual comments.
Tips:

- Before starting a data log upload in debug mode, clear the system logs
- Allow processes to stop and restart (such as after clearing logs, changing the upload file destination or rebooting the DAS) before diagnostics. For example, selecting "Apply" on the Setup / Upload page will stop the upload request processor on the DAS and restart it. This can take roughly 30 seconds.
- Use the browser page or tab refresh to update the logged information it displays. Some debug logs can be quite long, which can lug down the browser. Additionally, browser-based caching may even display stale pages.
- Watch for implied conditions from common messages, such as a SUCCESS message inside of a HTTP 500 http header. Same for non-script condition such as a dropped line.
- Once the scripts are working, don't forget to turn off debugging, and potentially clear those large system logs.

Many third party tools also exist, such as those for network traffic analysis and language compliance, which are beyond the scope of this document. Additionally, reference documents, language interpreter suppliers, or user groups on those topics may also provide useful information during diagnosis and debug.

Appendix A

Contents of the upload kit file

(Note that newer versions of the upload kit may contain changed content)

Directory	Notes
asp	Notes and examples of asp and IIS log file upload methods
documentation	Information about the upload kit and some related issues
DraftTableStructures	Some rough draft SQL device class notes
php	Example php scripts and notes
SampleConfig	Example configuration files from the DAS, and some example parsing code
SampleData	Example configuration and log files from the DAS
snmp	Sample of the formatted output using the UCD snmptrapd to log the AcquiSuite trap on a Linux system
tablestructure	Notes and examples of SQL table structures for MODBUSDEVICECLASS form variable http posts from the DAS
testforms	Some example html pages and some example server-client http conversations
wget	Example wget batch file scripts
xml	Example data sent from DAS in XML format