Overview of HealtheIntent Data Upload Utility (HIDUU)

Reference Pages

Exported on September 06, 2018

Published: June 23, 2017

Table of Contents

Published: June 23, 2017

Overview

HealtheIntent Data Upload Utility (HIDUU) enables you to upload different data sets into the *HealtheIntent* cloud-based system. HIDUU is intended to replace Data Upload Utility (DUU) as the primary means for uploading data in batch to the *HealtheIntent* system; for more information, see Benefits of *HealtheIntent* Data Upload Utility Over DUU.

The HIDUU is a *Java*-based command-line application that wraps Web services. It can be executed from either *Microsoft Windows* or *Unix* operating systems. It performs some file-level, client-side validation before uploading the file. OAuth is used to authorize the system from which the file is originating. The endpoint is protected by the *Cerner Care* OAuth service. You can achieve automation by scripting the execution of the utility's various commands.

Installation

See Installing *HealtheIntent* Data Upload Utility for installation instructions.

Pre-Upload Validation

To limit invalid files from being uploaded to the *HealtheIntent* system, HIDUU performs validation before attempting to upload the file. The type of validation depends on the command being executed. The results of the validation dictate how the utility proceeds with the upload; if the file fails the validation checks, the command does not perform the upload; however, if the file passes the validation checks with warnings, the command performs the upload.

Arguments Files

Arguments files can be set up for each command to store arguments that do not change frequently. Any argument relevant to a command can be added to the file. The argument file location is added as the -arg command. The system defaults any argument that is in both the command and file to the setting on the command.

Metadata

Each HIDUU upload command gathers metadata about the file being uploaded to the *HealtheIntent* system. The metadata is uploaded along with original file. Some of this metadata comes from command arguments, while others are harvested from the originating system. The following attributes are the metadata gathered by HIDUU:

Attribute	Meaning
record count	The value of the --*record-count* argument, if applicable. Used to suppress warnings when an empty file was intentionally uploaded.
upload reason	The value of the --*reason* argument, if applicable.
original file name	The value of the --*file* argument.
username	The username of the user logged into the system where the upload occurred.
utility version	The artifact and version of the upload utility.
upload ID	The GUID generated and assigned to the upload attempt by the utility.

Commands

The HIDUU contains the following commands:

- uploadDataSetFile
- uploadReferenceFile
- uploadUnvettedFile
- getDataSetInfo

HealtheIntent Data Upload Utility (HIDUU) Installation Instructions

Reference Pages

Exported on September 06, 2018

Published: August 09, 2018

Table of Contents

Published: August 09, 2018

System Requirements

Supported Operating Systems

HealtheIntent Data Upload Utility (HIDUU) can run on *Windows* and *UNIX* operating systems.

A minimum of 512 MB of RAM is required, and you must have a storage capacity large enough for the size of your extracts.

HIDUU generates a debugging log that causes a rollover if the file reaches a size of 1 GB.

Java Runtime Environment (JRE)

A *Java* 1.8 (*Java* 8) run-time environment (JRE) must be present to upload. The existence of a JAVA_HOME environment variable overrides the default mechanism of locating the JRE using PATH. To maintain explicit control over which JRE is selected, particularly if multiple JRE versions are installed on the host operating system, the Cerner standard is to define JAVA_HOME.

In most cases, the default settings of the JRE are sufficient for the upload. If certain aspects must be configured, such as controlling the heap size, you can pass options to the *Java* command using the JAVA_OPTS environment variable. For example, you can adjust the minimum and maximum heap size to 512 MB and 1024 MB using the following command:

On *UNIX*:

```
$ export JAVA_OPTS="-Xms512m -Xmx1024m"
```

On *Windows*:

```
$ set JAVA_OPTS=-Xms512m -Xmx1024m
```

Downloads

Clients

You can download HIDUU on the Distributions page.

Package Number	96608
Package Name	*HealtheIntent* Data Upload Utility (HIDUU)
HIDUU version	1.6
Wolfe Version	1.1

Cerner

Cerner associates can download HIDUU from the release repository.

Release Repository	http://repo.release.cerner.corp/nexus/content/repositories/main-repo/com/cerner/pophealth/midas/hi-data-upload-utility/

Installation

Unpack

Unzip the .OCD file and unzip the hi-data-upload-utility-1.6-bin.zip file within the .OCD .ZIP archive. This creates a hi-data-upload-utility-1.6 folder on your device.

Validation IP Addresses

The IP addresses below are used for authentication, validation, and file upload. Each address needs to be whitelisted for any firewalls between the uploading system and the *HealtheIntent* system.

U.S. UK

U.S. IP Address	U.S. DNS Entry	Reason
159.140.143.97	https://[tenant mnemonic].dataingestiontool.healtheintent.com/	Data source and file validation
159.140.206.15	https://api.cernercare.com/oauth/access	User authentication
159.140.207.48	https://api.data-collector.cernercentral.com/	File upload

U.S. UK

UK IP Address	UK DNS Entry	Reason
10.40.102.204	https://[tenant mnemonic].dataingestiontool.eu.healtheintent.com/api/da-validation-service/	Data source and file validation
10.40.102.114	https://oauth-api.eu.cerner.com/oauth/access	User authentication
10.40.102.211	https://data-collector-api.eu.cernercentral.com	File upload

Configure

Minimal configuration is required for HIDUU. As an added security feature, each client is provided with their own client-specific URL used by HIDUU to connect to *HealtheIntent*. Use the install.sh (*UNIX*) or install.bat (*Windows*) script to initialize the connection properties on your device using your *HealtheIntent* client mnemonic.Configure

Note
If you do not know your client mnemonic, contact the Cerner DataWorks team to obtain it.

On *UNIX*:

```
$ sh /hi-data-upload-utility-1.6/bin/install
```

On *Windows*:

```
$ \hi-data-upload-utility-1.6\bin\install.bat
```

If your client mnemonic is, for example, **myclientmnemonic**, the output would be displayed as in the following example:

```
$ sh /hi-data-upload-utility-1.6/bin/install.sh
Please enter Client Mnemonic:
myclientmnemonic
oauth-url=https://api.cernercare.com/oauth/access
validation-url=https://myclientmnemonic.dataingestiontool.healtheintent.com/
api/da-validation-service/
collector-url=https://api.data-collector.cernercentral.com/
connection.properties file was created!
```

You can find the connection Web addresses in the `/hi-data-upload-utility-1.6/lib/connection.properties` file.

Test Your Installation

The hi-data-upload-utility *UNIX* shell script that locates the *Java* run time and constructs the class path is provided in the /bin folder. The script adds all the .JAR files located in the /lib folder to the class path. Similarly, a hi-data-upload-utility.bat*Windows* batch file that achieves an equivalent objective is also provided. To run hi-data-upload-utility.bat from any folder on a *Windows* operating system, add the hi-data-upload-utility-1.6\bin folder to your local machine's PATH environment variable.

To test your installation on *UNIX*, execute the hi-data-upload-utility command with no parameters.

```
$ sh hi-data-upload-utility-1.6/bin/hi-data-upload-utility
```

If the downloaded .ZIP file was unpacked correctly, the output looks like the following example:

```
$ sh hi-data-upload-utility-1.6/bin/hi-data-upload-utility
[INFO] 2016-03-09 16:52:39,941: Usage: <command> <args>

Commands:
uploadDataSetFile Command for streaming a file from local file system to the
Healthe Intent cloud platform specifically for data assimilation (i.e.
normalization) processing.
uploadReferenceFile Command for streaming a reference file from local file
system to the Healthe Intent cloud platform specifically for data
assimilation (i.e. normalization) processing.
uploadUnvettedFile Command for streaming an unvetted file from local file
system to the Healthe Intent cloud platform specifically for data
assimilation (i.e. normalization) processing.
getDataSetInfo Command that downloads information on all data sets belonging
to all data sources associated with the specified system account from
HealtheIntent.
```

Note that this does not test the connection to *HealtheIntent*, only that the .ZIP file was unpackaged correctly, that the *Java* run time can be found, and that the class path was constructed correctly.

Automating HIDUU

Once data is being sent to *HealtheIntent* at regular intervals, you may want to establish a long-term, automated system that uses HIDUU to transmit files. Due to the variance in network infrastructure and resources across client sites, Cerner does not offer a prescriptive recommendation or standard for implementing HIDUU automation. If you are designing an automated system, however, it should include the following components:

- A shared, central location for delivering files to *HealtheIntent*
- A server containing the HIDUU application used to transmit the files
- Scripts that work in conjunction with a database or configuration file that provides the correct HIDUU parameters to use for a given file in the shared location
- Optionally, a location to archive files and log transmissions

You may also want to address the following questions for your site:

- What person or group owns and maintains the automation at your site?
- What operating environment does the automation run on?
    - Is new hardware required?
    - Is new software required?
    - What scripting languages are available in your operating environment?
    - What skills and training are required for additional personnel to support the automation?
- How do internal data owners deliver their extract files to the automation system?
    - What internal network security exists that might complicate this?
    - What file and directory permissions are required by the automation system?
- How does the automation system know which HIDUU parameters to use when a new file is delivered to the system for transmission to *HealtheIntent*?
    - How does it determine whether to send the files to the Live or Discovery environment?
- How does the automation system handle both successful and failed file transfers?
    - How does the system alert stakeholders?
    - How does the system handle retrying failed transfers?
- What triggers the automation system's scripts to determine which newly delivered files to send?
- How is logging and auditing handled?
- How are files archived?
- How does the system handle ad hoc requests to send a single file or resend an archived file?

See the *HealtheIntent* Community Space on *uCern Connect* for more information about how members of the *HealtheIntent* community have implemented the automation of *HIDUU*.

HIDUU Security FAQs

Are usernames and passwords transmitted using encryption?

The service uses an internal implementation of the two-legged OAuth 1.0a specification to authenticate uploads. Key and secret information is passed to the OAuth service using an HTTPS connection. No explicit encryption is performed by HIDUU.

Does the application or service encrypt data before sending it over the internet or an open network?

No, not directly, though data is transmitted using HTTPS protocol.

Does Cerner completely delete customer data when a customer deletes it from their web services?

HIDUU does not delete data, but you can request that raw data that has been uploaded be deleted from the Hadoop Distributed File System (HDFS).

Does Cerner review security during each phase of software development?

Security vulnerabilities are considered when reviewing any code changes. Code that is deployed is independently reviewed and verified by multiple team engineers and architects. Processes also are validated as having been followed during the development lifecycle as part of the code's release process.

Do third parties conduct security assessments for your application or service?

The service is penetration tested, both by internal security teams and external auditors

Does the application or service support SSL version 3 or TLS?

Secure Sockets Layer (SSL) is disabled. We support only Transport Layer Security (TLS) 1.0, TLS 1.1, and TLS 1.2. See the SSL report for the Data Collector API on the *Qualys* website for more information about the web server configuration.

Does the application use only IP ports 80 and 443 (HTTP and HTTPS)?

Yes.

This page includes links to external resources. These resources are provided for reference purposes and should be used with caution. Contact your Cerner support team for more information about third-party content.

Upload Data Set File Command Using HealtheIntent Data Upload Utility (HIDUU)

Reference Pages

Exported on September 06, 2018

Published: October 16, 2017

Table of Contents

Published: October 16, 2017

Overview

The Upload Data Set File command uploads a vetted data set file to *HealtheIntent*. In order for a file to become a vetted data set file, a sample file must first by analyzed by a Cerner data architect. Typically, unvetted sample files are uploaded using the HIDUU uploadUnvettedFile command. As part of the vetting process, the data architect collaborates with the data source owner to agree on a file specification. The file specification acts as an interface contract between the data source system and the *HealtheIntent* system. The file specification is used to develop the validation checks that the HIDUU performs prior to uploading or processing the data set file. It is also used to develop the mapping of the file to *HealtheIntent* data models.

With the inclusion or exclusion of a single command argument, the Upload Data Set File command can upload files to either the Live or Discovery environment. This provides a mechanism for integration testing the execution of the HIDUU command without actually having the uploaded files process into the Live environment.

Options also can be specified as command line arguments by recording them in an external file as --args.

Command Arguments

Short Arg	Long Arg	Required (R) or Optional (O)	Description
-said	--system-account-id	R	The system account ID used to authorize the system uploading the file.
-sas	--system-account-secret	R	The system account secret used to authorize the system uploading the file.
-sid	--source-id	R	The ID of the data source to which the data set file belongs.
-dsid	--data-set-id	R	The ID of the data set, within the data source, to which the file is assigned.
-fid	--file-id	R	The ID of the file within the specific data set specification version (for example, SINGLE_FILE, FILE_1, FILE_2, and so on).
-rl	--release	R	The release of the data set the file belongs to (for example, 20150701). Unless otherwise configured, the expected formats are yyyyMMdd or yyyyMMddHHmm. The release identifies all data set files that should be processed together for multifile data sets. The release dictates the order in which the data set is processed (for example, a release of 20150701 is processed before 20150702).
-sv	--spec-version	R	The data set specification version to which the file conforms.
-f	--file	R	The fully qualified path to the file to upload.
-ss	--sub-source	O	The subsource of the data source to which the file is assigned. A source must be configured as supporting subsources.
-re	--reason	O	The reason for uploading the file, which usually is included for ad hoc uploads. Enclose text in single quotes (') to ensure that the full text is sent.

Short Arg	Long Arg	Required (R) or Optional (O)	Description
-rc	--record-count	O	The number of records contained in the file. Particularly useful to communicate that zero records were expected in an empty file.
-d	--discovery	O	If this nonvalued argument is specified, the file is loaded to the Discovery (test) environment.
-a	--args	O	The fully-qualified path to the file with the default arguments for the command.
-e	--encryption	O	If this nonvalued argument is specified, the file being uploaded has been encrypted using Pretty Good Privacy (PGP) encryption.
-cm	--client-mnemonic	O	*ADVANCED USE ONLY* The *HealtheIntent* mnemonic of the client for whom the file is being uploaded. ⓘ **Note** This command argument is labeled for advanced use only because it overwrites a value in the Configuration file and should be used only by Cerner associates submitting files for clients.

Data Set File Composite Identifier

A data set file is uniquely identified by combining several file attributes, each of which is specified as a command argument. Together these attributes form a composite identifier that uniquely identifies the data set file in *HealtheIntent*. A file uploaded with the same composite identifier as a previously uploaded file is considered a new version, and therefore it replaces the previous version.

The following attributes make up the data set file composite identifier:

- source-id
- data-set-id
- file-id
- release
- spec-version
- sub-source (optional)

Uploading Analytics Data

The Upload Data Set File command can be used to upload *HealtheAnalytics* files. If you are transitioning from using the Data Upload Utility (DUU), you must make the following adjustments:

- Datatypes in DUU is called data-set-id in HIDUU.
- The spec-version argument for files being uploaded for *HealtheAnalytics* should always be set to **1**.
- The file-id argument for files being uploaded for *HealtheAnalytics* should always be set to **SINGLE_FILE**.

Preupload Validation

HealtheIntent Data Upload Utility (HIDUU) performs the following validation before uploading a file using the Data Set File command:

- Valid release format (for example, yyyyMMdd)
- Valid source-id, data-set-id, spec-version, and file-id combination
- Header record validation for .CSV files
- First record length validation for fixed-width files

Release Format

The release portion of the composite identifier dictates the order in which the received data sets are processed. This allows correct processing of data set releases, even when releases are received out of order. The release values are lexicographically compared for ordering purposes, therefore it is important that the release values are consistent across all releases of the data set.

In is strongly recommended that you use one of the following two release format options for all new data sets:

- yyyyMMdd
- yyyyMMddHHmm

To remain passive, existing data sets can be configured to accept one of the following release formats:

- yyyyMMddhhmmss (for example, 20151005131415)
- yyyyMMddhhmmssSS (for example, 2015100513141516)
- yyyyMMddThhmmss (for example, 20151005T131415)

In the rare case that an existing data set currently use a format that is not listed above, the release format validation can be turned off for that data set. It is highly recommended that this option be used as a last resort.

Return Status

HIDUU displays the message **File successfully uploaded** if the file is uploaded successfully.

HIDUU displays the message **File failed to upload** if the file is not uploaded successfully.

See the View the Exit Status of Upload Commands page for more information about viewing the exit status of a command.

Sample Commands

Sample

```
hi-data-upload-utility uploadDataSetFile -said 2ad2874c-18ee-4f46-8d09-3956525672135 -sas
adfa************************adfa -sid 425d874c-18ee-4f46-8d09-s43fas56763d -dsid
MAY_TX_ALLSCRIPTS_EMR_ALLERGY -sv 1 -fid SINGLE_FILE -rl 20150103 -f /raw/allscriptsallergy_20150103.csv -
re 'testing hiduu'
```

Sample with Analytics File

```
hi-data-upload-utility uploadDataSetFile -said 2ad2874c-18ee-4f46-8d09-3956525672135 -sas
adfa************************adfa -sid 425d874c-18ee-4f46-8d09-s43fas56763d -dsid MAY_TX_ANALYTICS -sv 1 -
fid SINGLE_FILE -rl 20150203 -f /surveys/satisfaction_2014Q4.csv -re 'testing hiduu Analytics'
```

Sample with Argument File

```
hi-data-upload-utility uploadDataSetFile -rl 20150103 -arg /arg/dataset.txt -re 'testing hiduu Argument'
```

Troubleshoot HealtheIntent Data Upload Utility (HIDUU)

Reference Pages

Exported on September 06, 2018

Published: September 27, 2017

Table of Contents

Published: September 27, 2017

Several options need to be specified accurately in order for *HealtheIntent* Data Upload Utility (HIDUU) to upload data. If you omit required options, enter invalid options, provide entity types that do not conform to your specification, or perform any of the other actions listed on the table below, the upload terminates and an error message is displayed. See the table below for explanations and resolutions for the error messages.

Note
You can find additional information for each error message in the log folder for the directory in which the command is run.

Error Message	Explanation	Resolution
Argument [File-id] had an invalid value [xxxxx].	The file ID argument value is incorrect.	Verify that the file ID argument value is valid.
Argument [Record-count] had an invalid value [xxxxx]. Value must be a positive integer.	The record-count argument value must be a positive integer.	Carefully review the record-count argument and correct the value.
Dataset ID not found.	The dataset ID is not configured in the source registry.	Verify that the dataset ID is correct and configured in the source registry.
Error reading [File name]: [Error message].	HIDUU is not able to read the file being uploaded for pre-upload verification.	Carefully review the file being uploaded and the folders in which it is located.

Error Message	Explanation	Resolution

Error while trying to reach the validation services: [Unable to invoke request].	The system was unable to connect to the correct URL to upload the file.	Check the **connection.properties** file in the **lib** directory. Ensure that the *HealtheIntent* Client Mnemonic is populated and the URL is correct.
		See the following example of a connection.properties file:
		`client= [CLIENT_MNEMONIC]`
		`oauth-url=https://api.cernercare.com/oauth/access`
		`validation-url=https://[CLIENT_MNEMONIC].dataingestiontool.healtheintent.com/api/da-validation-service/`
		`collector-url=https://api.data-collector.cernercentral.com/`

File ID not found.	The file ID is incorrect for this source or dataset ID.	Verify that the file ID argument is correct.
Header does not match expected.	The header in the file does not match the configured header.	Verify that you are sending the file with the correct file ID option.
Missing required argument [Argument name].	The argument listed is included but no value set exists for it.	Carefully review the argument and either add a value set or correct it.
No schema validation strategy defined for file.	The mapping configuration, which includes the file header, has not been deployed, so the header record cannot be validated. Files without deployed mapping configurations are only allowed to be uploaded to the Discovery environment.	Verify that the mapping configuration is deployed to the environment.

Error Message	Explanation	Resolution
Release date format does not match.	The date format of the release argument does not conform to the allowed formats for HIDUU.	Update the date format for the release argument to conform to the allowed release formats. Unless otherwise configured, the expected formats are yyyyMMdd or yyyyMMddHHmm.
Release date format does not match expected.	The date format of the release argument conforms to the allowed formats but is not specified in the source registry.	Check the source registry for the valid date and time formats.
Source ID '[xxxxxxxxx]' could not be found in environment 'LIVE'.	The source ID is not configured in the source registry.	Verify that the source ID is correct and configured in the source registry.
Subsource is required.	The source ID specified requires the subsource argument to be included, but the command run did not include a subsource argument.	Verify that the source ID is correct. If it is correct, add a subsource argument to the command.
System account authentication error.	This error usually indicates that the OAuth secret or key specified using --oauth-secret or --oauth-key is incorrect. The cause is typically due to a typographical error, which results in a token signature that cannot be verified by the collector service.	Carefully review the OAuth secret to ensure that no typographical errors exist. OAuth secrets are case-sensitive.
The service encountered an unexpected condition which prevented it from fulfilling the request: [Service error].	HIDUU is not able to complete the upload because of an undefined condition.	Upload the data or file again. If the error message is displayed, contact your Cerner support team.

View the Exit Status of Upload Commands

Reference Pages

Exported on September 06, 2018

Published: September 16, 2016

Table of Contents

Published: September 16, 2016

Overview

Every upload command in *HealtheIntent* Data Upload Utility (HIDUU) displays an exit status, which is sometimes referred to as a return status or exit code. A successful upload command displays a **0**, while an unsuccessful upload command displays a **1**.

You can either use the echo $? command to view the exit status of a single upload command, or you can edit the HIDUU code in *Linux* or *Unix* to configure the system to display the exit statuses of all future upload commands.

Use the Echo $? Command to View the Exit Status for a Single Upload Command

To view the exit status for a single upload command using the echo $? command, enter **echo $?** after the hi-data-upload-utility upload command terminates.

Examples

The example below shows what is displayed for a successful file upload when you use the echo $? command.

```
➜ ./hi-data-upload-utility uploadReferenceFile -sas K34xQ*******************yr3ta -sid e36f10f1-4f8d-48dc-
b9e8-ecba44b1c28e -said e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -reference-file-name test -f text.txt
[INFO] 2016-08-28T14:36:18,941: system-account-secret     : *******************
[INFO] 2016-08-28T14:36:18,941: source-id                 : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-08-28T14:36:18,941: system-account-id         : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-08-28T14:36:18,941: reference-file-name       : test
[INFO] 2016-08-28T14:36:18,942: file                      : text.txt
[INFO] 2016-08-28T14:36:21,785: File successfully uploaded.
➜  bin git:(master) ✗ echo $?
0
```

The example below shows what is displayed for a failed file upload when you use the echo $? command.

```
./hi-data-upload-utility uploadReferenceFile -sas K34xQ*******************yr3ta -sid e36f10f1-4f8d-48dc-
b9e8-ecba44b1c28e -said e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -reference-file-name test -f text.txt -d
[INFO] 2016-08-28T14:35:34,272: system-account-secret     : *******************
[INFO] 2016-08-28T14:35:34,272: source-id                 : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-08-28T14:35:34,273: system-account-id         : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-08-28T14:35:34,273: reference-file-name       : test
[INFO] 2016-08-28T14:35:34,273: file                      : text.txt
[INFO] 2016-08-28T14:35:34,273: discovery                 : True
[ERROR] 2016-08-28T14:35:36,750: Consumer key 'e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e' does not have read
access to client 'xxxxx' with id '6eff09af-92bc-4118-9b99-528f31c27326'. Check the policy for the
application to verify that your consumer key is authorized.
[INFO] 2016-08-28T14:35:36,753: File failed to upload.
➜  bin git:(master) ✗ echo $?
1
```

Add Code to View the Exit Status

As an alternative to using the echo $? command to view individual upload commands, you can edit the HIDUU code in either *Linux* or *Unix* to configure the system to display the exit statuses of all future upload commands.

> **Important!**
> This option should only be used by advanced HIDUU users. Adding the code incorrectly can cause HIDUU to stop functioning. Use the echo $? command described in the Use the Echo $? Command to View the Exit Status for a Single Upload Command section if you do not want to edit the HIDUU code directly.

Add Code in *Linux*

To modify HIDUU to display exit statuses for all upload commands in *Linux*, add the code below to the hi-data-upload-utility.sh script in the location in the example.

```
if [ $? -eq 0 ]; then
  echo Command executed successfully - "Status 0"
else
  echo Command failed to execute - "Status " $?
fi
```

Example

```
#!/bin/bash
if [[ -z "$JAVA_HOME" ]]; then
    __JAVA=$(which java)
    if [[ -z "$__JAVA" ]]; then
        echo "no java runtime found; define JAVA_HOME or put java in PATH"
        exit 1
    fi
else
    __JAVA=$JAVA_HOME/bin/java
    if [[ ! (-f "$__JAVA" && -x "$__JAVA") ]]; then
        echo "$__JAVA: not found; verify JAVA_HOME environment variable"
        exit 1
    fi
fi
__BASE_DIR=$(cd $(dirname $0)/..; pwd)
__CLASSPATH=$(echo $__BASE_DIR/lib/*.jar | sed "s/ /:/g"):$__BASE_DIR/lib
__CLASS=com.cerner.pophealth.hiduu.HiDuuMain
__DEBUG=-agentlib:jdwp=transport=dt_socket,address=8081,server=y,suspend=y
$__JAVA $JAVA_OPTS  -classpath $__CLASSPATH $__CLASS "$@"

if [ $? -eq 0 ]; then
  echo Command executed successfully - "Status 0"
```

```
else
  echo Command failed to execute - "Status " $?
fi
```

Add Code in *Unix*

To modify HIDUU to display exit statuses for all upload commands in *Unix*, add the code below to the hi-data-upload-utility.sh script in the location in the example.

```
if %errorlevel% == 0 (
  echo Command executed successfully - Status 0
) else (
  echo Command failed to execute - Status %errorlevel%
)
```

Example

```
@echo off
if "%JAVA_HOME%" == "" (
  where /q java.exe
  if %errorlevel% == 0 (
    set __JAVA=java.exe
    goto :buildClasspath
  ) else (
    echo no java runtime found; define JAVA_HOME or put java.exe in PATH
    goto :error
  )
)
set __JAVA="%JAVA_HOME%\bin\java.exe"
if not exist %__JAVA% (
  echo %__JAVA%: not found; verify JAVA_HOME environment variable
  goto :error
)
:buildClasspath
for %%P in (%~dp0\..) do set __BASE_DIR=%%~fP
set __CLASSPATH=
for /r "%__BASE_DIR%\lib" %%P in (*.jar) do call :addToClasspath %%P
set __CLASSPATH=%__CLASSPATH%;%__BASE_DIR%\lib
goto :run
:addToClasspath
set __CLASSPATH=%__CLASSPATH%;"%1"
goto :eof
:run
%__JAVA% %JAVA_OPTS% -classpath %__CLASSPATH% com.cerner.pophealth.hiduu.HiDuuMain %*
if %errorlevel% == 0 (
  echo Command executed successfully - Status 0
) else (
  echo Command failed to execute - Status %errorlevel%
)
call :cleanup
exit /b %errorlevel%
:error
call :cleanup
exit /b 1
:cleanup
set __JAVA=
set __BASE_DIR=
set __CLASSPATH=
goto :eof%
```

Examples

The example below shows the exit status that is displayed for successful file uploads when you have edited the HIDUU code.

```
➔ ./hi-data-upload-utility uploadDADataSetFile  -sas K3***********************ftyr3ta -sid
e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -said e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -f /Users/jg010252/
Downloads/_data_acquisition_tmp_validation-mappings/DOUBLE_CSV_VALIDATION/file2_withHeader.csv -dsid
DOUBLE_CSV_VALIDATION -fid FILE_2 -rl 20150103 -sv 1
[INFO] 2016-03-22 17:40:16,159: system-account-secret     : *******************
[INFO] 2016-03-22 17:40:16,159: source-id                 : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-03-22 17:40:16,160: system-account-id         : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-03-22 17:40:16,160: file                      : /Users/jg010252/Downloads/
_data_acquisition_tmp_validation-mappings/DOUBLE_CSV_VALIDATION/file2_withHeader.csv
[INFO] 2016-03-22 17:40:16,160: data-set-id               : DOUBLE_CSV_VALIDATION
[INFO] 2016-03-22 17:40:16,160: file-id                   : FILE_2
[INFO] 2016-03-22 17:40:16,160: release                   : 20150103
[INFO] 2016-03-22 17:40:16,160: spec-version              : 1
[ERROR] 2016-03-22 17:40:17,534: Upload key was valid, but this release has already been uploaded.
[INFO] 2016-03-22 17:40:18,721: [jg010252] successfully uploaded file [/Users/jg010252/Downloads/
_data_acquisition_tmp_validation-mappings/DOUBLE_CSV_VALIDATION/file2_withHeader.csv].
[INFO] 2016-03-22 17:40:19,039: File successfully uploaded.
Command executed successfully - Status 0
```

The example below shows the exit status that is displayed for failed file uploads when you have edited the HIDUU code.

```
➔ ./hi-data-upload-utility uploadDADataSetFile  -sas K3***********************ftyr3ta -sid
e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -said e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e -f /Users/jg010252/
Downloads/_data_acquisition_tmp_validation-mappings/DOUBLE_CSV_VALIDATION/file2_withHeader.csv -dsid
DOUBLE_CSV_VALIDATION -fid FILE_2 -rl 20150103 -sv 1:
[INFO] 2016-03-22 17:40:36,060: system-account-secret     : *******************
[INFO] 2016-03-22 17:40:36,061: source-id                 : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
[INFO] 2016-03-22 17:40:36,061: system-account-id         : e36f10f1-4f8d-48dc-b9e8-ecba44b1c28e
```

```
[INFO] 2016-03-22 17:40:36,061: file                      : /Users/jg010252/Downloads/
_data_acquisition_tmp_validation-mappings/DOUBLE_CSV_VALIDATION/file2_withHeader.csv
[INFO] 2016-03-22 17:40:36,061: data-set-id               : DOUBLE_CSV_VALIDATION
[INFO] 2016-03-22 17:40:36,061: file-id                   : FILE_2
[INFO] 2016-03-22 17:40:36,061: release                   : 20150103
[INFO] 2016-03-22 17:40:36,061: spec-version              : 1:
[ERROR] 2016-03-22 17:40:36,629: Argument [spec-version] had an illegal character. This argument can only
contain letters, numbers, underscores, and hyphens. Please recheck the entered value [1:].
[ERROR] 2016-03-22 17:40:36,629: Error building upload key.
[ERROR] 2016-03-22 17:40:36,629: File failed to upload.
Command failed to execute - Status  1
```