AMPS Utilities Guide

AMPS Utilities Guide

5.0

Publication date Jan 19, 2017

Table of Contents

Chapter 1. Utilities

AMPS provides several utilities that are not essential to message processing, but can be helpful in troubleshooting or tuning an AMPS instance:

- `amps_sow_dump` is used to inspect the contents of a SOW topic store.
- `amps_journal_dump` is used to examine the contents of an AMPS journal file during debugging and program tuning.
- `ampserr` is used to expand and examine error messages that may be observed in the logs. This utility allows a user to input a specific error code, or a class of error codes, examine the error message in more detail, and where applicable, view known solutions to similar issues.
- AMPS provides a command-line Spark client as a useful tool for checking the status of the AMPS engine. The Spark client can also be used to run queries, place subscriptions, and publish data.
- `amps_upgrade` upgrades data files for existing AMPS instances to the current release of AMPS.

Chapter 2. amps_sow_dump

`amps_sow_dump` is a utility used to inspect the contents of a SOW topic store. Additionally it can be used to gather summary statistics on a SOW file.

2.1. Options and Parameters

Table 2.1. Parameters for `amps_sow_dump`

Option	Description
`filename`	Filename of the SOW file.
-n LIMIT	Maximum number of records to print per file.
-v, --verbose	Print record metadata for records and file summary.
--sizing-chart	Print memory sizing chart for efficiency comparison (experimental).
-e, --escape	Escape special characters in record data and header.
-d DELIMITER	Prints only the record data using the provided ASCII character value as the record delimiter [default: 10 for newline].
--version	Show the version number of the program and exit.
-h, --help	Show the help message and exit.

2.2. Usage

Example 2.1 shows a simple sow dump with the `-e` flag set to make the header, message and field separators readable. Each key which exists in the `order.sow` file is dumped out to stdout. This output can easily be redirected to a new file, or piped into another program for further analysis.

This example also uses the `-e` flag which escapes the special characters. The purpose of this is to simplify the output presented. The field separator used in this example is `byte 1` which is replaced with as `\x01` by the `-e` flag.. If this flag was not used, then a non-ascii character would be displayed, making the output harder to read.

> `amps_sow_dump` expects a filename at a minimum in order to complete the SOW topic store dump process.

```
%> ./amps_sow_dump -e ./order.sow

id=0\x01value=1743\x01
id=1\x01value=6554\x01
id=2\x01value=3243\x01
id=3\x01value=5332\x01
id=4\x01value=3725\x01
id=5\x01value=1598\x01
id=6\x01value=6094\x01
id=7\x01value=7524\x01
```

```
id=8\x01value=2432\x01
id=9\x01value=9669\x01
id=10\x01value=140\x01
```

Example 2.1. Example of amps_sow_dump Output

2.3. Verbose Output

The `amps_sow_dump` utility also provides for verbose output, which will display more information about the file and its structure in addition to the records contained in the file.

```
❶key               = 13480918659780819530
 crc               = 3156974437
 flags             = 0
 file offset       = 4352
 slab offset       = 4096
 allocated         = 128
 data size         = 21
 expiration        = 0
 update            = 0
 generation        = 0
 seq               = 0
 data              = [1=10001
 2=aaaaaaaaaa
 ]

❷File                : ./sow/order.sow
 Version             : amps-sow-v1.0
 Valid Keys          :                10000
 Record Size         :                  512
 Maximum Records     :                10000
 Multirecords        :                    0
 Maximum record size :                   21
 Average record size :                21.00
 Slab Count          :                    1

 Slab Detail
❸size          :             5128192
 file offset   :                4096
 valid count   :               10000
 invalid count :                   0
 stored bytes  :             1280000
 data bytes    :              210000
 deleted bytes :                   0
```

Example 2.2. amps_sow_dump verbose output

❶ This is the last record reported by `amps_sow_dump` for this sample SOW file. Table 2.2 describes the rows in this record.

Table 2.2. Parameters for a record in `amps_sow_dump`

Option	Description
key	This is the `SOWKey` for this record - a unique identifier used by AMPS clients to identify a record.
crc	The error checking value used to verify the data integrity of the record.
flags	A mask used to identify settings which have been triggered by the AMPS SOW file store for maintaining this record.
file offset	This is the location within the file for the record.
slab offset	The location within the slab for the record.
allocated	S The number of bytes allocated for the record.
data size	The size, in bytes of the data contained in the record.
expiration	If set to a value greater than 0, this represents he timestamp when the record will expire from the SOW.
update	Count of updates sent to the record.
generation	Each update increments the generation count for a SOW record.
seq	The sequence passed in with the message and stored with the record.
data	The message data stored in the SOW record.

❷

Table 2.3. Parameters for a file in `amps_sow_dump`

Option	Description
File	The filesystem location where the SOW records are persisted.
Version	The AMPS SOW file format version.
Valid Keys	The number of unique records persisted in the SOW file.
Record Size	The number of bytes allocated for each record.
Maximum Records	The maximum number of records stored in the SOW file.
Multirecords	The number of records which whose contents are larger than the `Record Size` and require nultiple records to store the data.
Maximum Record Size	The size of the largest record persisted in the SOW file.
Average Record Size	The average size of all records stored in the SOW file.
Slab Count	The number of slabs allocated to the SOW file.

❸

2.4. Sizing Chart

Example 2.3 shows the output from the `--sizing-chart` flag. This is feature can be useful in tuning AMPS memory usage and performance. The `Record Size` with the asterisk shows the current `Record Size` setting and allows an AMPS administrator to compare memory usage efficiency along with the potential for a multi-record penalty.

> This feature is currently listed as experimental, so changing AMPS record size configuration based on the results may not necessarily help performance, and could hurt performance in some cases.

```
%> ./amps_sow_dump --sizing-chart ./order.sow

================================================================
   Record Size           Store     Efficiency   Multirecords
================================================================
 128           128 B        100.00%              0
 256           256 B         50.00%              0
 384           384 B         33.33%              0
 512*          512 B         25.00%              0
 640           640 B         20.00%              0
 768           768 B         16.67%              0
 896           896 B         14.29%              0
1024          1024 B         12.50%              0
1152       1.12 KB           11.11%              0
1280       1.25 KB           10.00%              0
1408       1.38 KB            9.09%              0
1536       1.50 KB            8.33%              0
1664       1.62 KB            7.69%              0
1792       1.75 KB            7.14%              0
1920       1.88 KB            6.67%              0
```

Example 2.3. Example Output for `--sizing-chart`

Chapter 3. amps_journal_dump

The AMPS journal dump utility is used in examining the contents of an AMPS journal file for debugging and program tuning. The `amps_journal_dump` utility is most commonly used as a tool to debug the forensic lifespan of messages that have previously been published to AMPS. The `amps_journal_dump` tool is used to show that messages exist in a journaled topic, and to show the order the message was received in, and the timestamp associated with the message.

3.1. Command Line Options

The `amps_journal_dump` program has the following options available. These can also be printed to the screen by typing `amps_journal_dump -help`.

Table 3.1. Parameters for `amps_journal_dump`

Option	Description
`filename`	Filename of the AMPS journal file..
--version	Show the program version number and exit.
-h, --help	Show the program help message and quit.
-l LIMIT	Limit range of output to entris N:M where N is the first entry and M is the last entry. Passing in a single value, M, will return the first M results.

3.2. Looking at the Output

In this section will examine some sample output from running `amps_journal_dump`. We will then go over what each of the entries emitted by the program means.

```
File Name  : ./AMPS.0000000000.journal
File Size  : 10485760
Version    :       amps::txlog/vx
Extents    :               [1:10]
_____________________________________________
Entry              :                        5644
CRC                :                  3483899014
type               :                     publish
entry size         :                         512
msg len            :                          11
msg type           :                         fix
localTxId          :                        5645
sourceTxId         :                           0
source             :                           0
client             :       13683435528643874114
clientSeq          :                        1353
topicHash          :       10864674256041732524
SOW     Key        :       18446744073709551615
```

```
iso8601 timestamp   :      20130612T151247.410553
flags               :                          0
topic len           :                         40
auth ID len         :                          0
topic               : [test_topic_name]
auth ID             : []
data                : [1=1353 2=a ]


-------------------------------------------
Total Entries    : 1
Total Bytes      : 5632
Remaining Bytes  : 10480128
```

Example 3.1. Example of amps_journal_dump Output

As is apparent in Example 3.1, the output from `amps_journal_dump` is split into three sections, a header, a listing of the contents of the journal file and a footer.

The header contains general information about the journal file as it is represented in the filesystem, and state data about the journal file.

Table 3.2. `amps_journal_dump` listing header.

Option	Description
File Name	The name fo the file as it appears on the local filesystem.
File Size	Number of total bytes allocated by the journal file.
Version	This is the version of the formatting used to write the data in the journal file.
Extents	A pair of numbers where the first is the number of extents used by the journal file, and the second is the number of blocks allocated for the journal file.

The second section of the `amps_journal_dump` lists each of the entries contained in the journal file, along with all of the meta-data used to track and describe the entry. For the sake of simplicity,Example 3.1 only shows a single listing, but it is more likely that a journal will contain multiple entries.

Table 3.3. `amps_journal_dump` sample listing

Option	Description
Entry	A monotonically increasing value representing the order in which the record was inserted into the transaction log file.
CRC	The cyclic redundancy check used for error checking the message.
type	The AMPS command used in the original message.
entry size	The number of bytes allocated to the transaction log record.
msg len	The number of bytes consumed by the `data` segment of the record.
msg type	The message type used to format the data segment in the record.
localTxId	The monotonically increasing identifier used across all records local transaction log journal files.
sourceTxId	The `localTxId` as it appears on the upstream replication source.
source	A unique identifier used to represent the upstream source of the record. If the source is `0` then the transaction originated from the current host.

Option	Description
client	The unique identifier associated with the client that published the message recorded in the transaction log.
clientSeq	The monotonically increasing sequence identifier from the client.
topicHash	The unique identifier for the topic the record was published to.
SOW Key	The unique identifer for the record in the SOW Topic.
iso 8601 timestamp	The ISO-8601 formatted timestamp representing the time the record was published to the transaction log. Notice that to keep timestamps consistent across instances that may be geographically dispersed, AMPS always timestamps in the UTC timezone. You can, however, ask AMPS to convert timestamps to a different timezone as described in Section 3.3.
timestamp	The raw timestamp stored in the AMPS transaction log. This is a microsecond-precision timestamp.
flags	A bitmask used to represent any set flags on the transaction log record.
topic len	The number of characters in the topic field.
auth ID len	The number of characters in the authId field.
topic	The plaintext name of the topic the record was published to.
authID	The identifier associated with the authentication token used with the publish message.
data	The raw data contained in the message.

As seen in Example 3.1, the final section contains general usage information about the data contained in the journal file.

Table 3.4. `amps_journal_dump` listing header.

Option	Description
Total Entries	Total number of journal entries entered into the journal file.
Total bytes	The number of reserved bytes consumed by the journal file.
Remaining Bytes	The number of unused bytes available out of the total reserved file size.

3.3. Timestamp formatting

The timestamp format used in `amps_journal_dump` is formatted by default using the system timezone for its location. To display the time in another timezone, the `TZ` environment variable can be confgired to modify thie output.

```
%> TZ='America/New_York' ./amps_journal_dump A.000000000.journal
```

Example 3.2. Example of formatting timestamp output for Eastern Timezone.

```
%> TZ='Asia/Tokyo' ./amps_journal_dump A.000000000.journal
```

Example 3.3. Example of formatting timestamp output for Tokyo.

```
%> TZ='Europe/London' ./amps_journal_dump A.000000000.journal
```

Example 3.4. Example of formatting timestamp output for London.

NOTE: This will not work on dates prior to 1970.

Chapter 4. ampserr

AMPS contains a utility to expand and examine error messages which may be observed in the logs. The `ampserr` utility allows a user to input a specific error code, or a class of error codes, examine the error message in more detail and, where applicable, view known solutions to similar issues.

4.1. Options and Parameters

Table 4.1. Parameters for `ampserr`

Option	Description
`error`	The error code to lookup. This can also be a regular expression.

4.2. Usage

The following example shows the output of the “00-0001” error message:

```
%> ./ampserr 01-0001
AMPS Message 00-0001 [level = info]

    DESCRIPTION: AMPS Copyright message.
    ACTION: No recommended action available.

Found 1 error matching '00-0001'.
```

Example 4.1. Example of ampserr Usage

The following example will return all messages that begin with “00-”. NOTE: For the sake of brevity, this manual does not include all messages that match this query.

```
%> ./ampserr 00-

AMPS Message 00-0000 [level = trace]

    DESCRIPTION: Internal log message used by AMPS
        development team. If you see this message
        logged, please notify AMPS support.

        ACTION: No recommended action available.

AMPS Message 30-0000 [level = warning]

    DESCRIPTION : AMPS internal thread monitoring has
        detected a thread that hasn’t made progress
        and appears 'stuck'. This can happen with long
        operations or a bug within AMPS.
```

```
    ACTION : Monitor AMPS and if these 'stuck'
        messages continue, then a restart of the engine
        could be the only way to resolve it. If it
        appears busy (high CPU utilization) then it
        could be a long operation (large query filter.)
```

Example 4.2. ampserr Usage with Regular Expression

The following example will return all error messages. NOTE: For the sake of brevity, this manual does not include all messages that match this query.

```
%> ./ampserr .

AMPS Message 00-0000 [level = trace]

    DESCRIPTION: Internal log message used by AMPS
        development team. If you see this message
        logged, please notify AMPS support.

    ACTION  No recommended action available.

AMPS Message 30-0000 [level = warning]

    DESCRIPTION : AMPS internal thread monitoring
        has detected a thread that hasn't made
        progress and appears 'stuck'. This can
        happen with long operations or a bug
        within AMPS.

    ACTION : Monitor AMPS and if these 'stuck'
        messages continue, then a restart of the
        engine could be the only way to resolve it.
        If it appears busy (high CPU utilization)
        then it could be a long operation (large
        query filter.)
```

Example 4.3. ampserr Usage with Regular Expression for Error Messages

Chapter 5. Spark

AMPS contains a command-line client, `spark`, which can be used to run queries, place subscriptions, and publish data. While it can be used for each of these purposes, `spark` is provided as a useful tool for informal testing and troubleshooting of AMPS instances. For example, you can use `spark` to test whether an AMPS instance is reachable from a particular system, or use `spark` to perform *ad hoc* queries to inspect the data in AMPS.

This chapter describes the commands available in the `spark`. For more information on the features available in AMPS, see the relevant chapters in the *AMPS User Guide*.

The `spark` utility is included in the `bin` directory of the AMPS install location. The `spark` client is written in Java, so running `spark` requires a Java Virtual Machine for Java 1.6 or later.

To run this client, simply type `./bin/spark` at the command line from the AMPS installation directory. AMPS will output the help screen as shown below, with a brief description of the `spark` client features.

```
%> ./bin/spark
================================
- Spark - AMPS client utility -
================================
Usage:

    spark help [command]

Supported Commands:

    help
    ping
    publish
    sow
    sow_and_subscribe
    sow_delete
    subscribe

Example:

    %> ./spark help sow

Returns the help and usage information for the 'sow' command.
```

Example 5.1. Spark Usage Screen

5.1. Getting help with spark

Spark requires that a supported command is passed as an argument. Within each supported command, there are additional unique requirements and options available to change the behavior of Spark and how it interacts with the AMPS engine.

For example, if more information was needed to run a `publish` command in Spark, the following would display the help screen for the Spark client's `publish` feature.

```
%>./spark help publish
==============================
- Spark - AMPS client utility -
==============================
Usage:

  spark publish [options]

Required Parameters:

  server    -- AMPS server to connect to
  topic     -- topic to publish to

Options:

  authenticator -- Custom AMPS authenticator factory to use
  delimiter     -- decimal value of message separator character
                   (default 10)
  delta         -- use delta publish
  file          -- file to publish records from, standard in when omitted
  proto         -- protocol to use (amps, fix, nvfix, xml)
             (type, prot are synonyms for backward compatibility)
                   (default: amps)
  rate          -- decimal value used to send messages
             at a fixed rate.  '.25' implies 1 message every
                   4 seconds. '1000' implies 1000 messages per second.

Example:

  % ./spark publish -server localhost:9003 -topic Trades -file data.fix

    Connects to the AMPS instance listening on port 9003 and publishes
 records
    found in the 'data.fix' file to topic 'Trades'.
```

Example 5.2. Usage of spark publish Command

5.2. Spark Commands

Below, the commands supported by `spark` will be shown, along with some examples of how to use the various commands and descriptions of the most commonly-used options. For the full range of options provided by `spark`, including options provided for compatibility with previous `spark` releases, use the `spark help` command as described above.

publish

The `publish` command is used to publish data to a topic on an AMPS server.

Common Options - spark publish

Table 5.1. Spark publish options

Option	Definition
`server`	AMPS server to connect to.
`topic`	Topic to publish to.
delimiter	Decimal value of message separator character (default 10).
delta	Use delta publish (sends a `delta_publish` command to AMPS).
file	File to publish messages from, stdin when omitted. `spark` interprets each line in the input as a message. The file provided to this argument can be either uncompressed or compressed in ZIP format.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.
rate	Messages to publish per second. This is a decimal value, so values less than 1 can be provided to create a delay of more than a second between messages. '.25' implies 1 message every 4 seconds. '1000' implies 1000 messages per second.
type	For protocols and transports that accept multiple message types on a given transport, specifies the message type to use.

Examples

The examples in this guide will demonstrate how to publish records to AMPS using the `spark` client in one of the three following ways: a single record, a python script or by file.

```
%> echo '{ "id" : 1, "data": "hello, world!" }' |  \
   ./spark publish -server localhost:9007 -type json -topic order

   total messages published: 1 (50.00/s)
```

Example 5.3. Publishing a single XML message.

In Example 5.3 a single record is published to AMPS using the `echo` command. If you are comfortable with creating records by hand this is a simple and effective way to test publishing in AMPS.

In the example, the JSON message is published to the topic *order* on the AMPS instance. This publish can be followed with a `sow` command in `spark` to test if the record was indeed published to the *order*topic.

```
%> python -c "for n in xrange(100): print '{\"id\":%d}' % n" | \
   ./spark publish -topic disorder -type json -rate 50 \
   -server localhost:9007

   total messages published: 100 (50.00/s)
```

Example 5.4. Publishing multiple messages using python.

In Example 5.4 the `-c` flag is used to pass in a simple loop and print command to the python interpreter and have it print the results to `stdout`.

The python script generates 100 JSON messages of the form `{"id":0}`, `{"id":1}` ... `{"id":99}`. The output of this command is then *piped* to spark using the | character, which will publish the messages to the *disorder* topic inside the AMPS instance.

```
%> ./spark publish -server localhost:9007 -type json -topic chaos \
   -file data.json

   total messages published: 50 (12000.00/s)
```

Example 5.5. Spark publish from a file

Generating a file of test data is a common way to test AMPS functionality. Example 5.5 demonstrates how to publish a file of data to the topic *chaos* in an AMPS server. As mentioned above, `spark` interprets each line of the file as a distinct message.

sow

The `sow` command allows a `spark` client to query the latest messages which have been persisted to a topic. The SOW in AMPS acts as a database last update cache, and the `sow` command in `spark` is one of the ways to query the database. This `sow` command supports regular expression topic matching and content filtering, which allow a query to be very specific when looking for data.

For the `sow` command to succeed, the topic queried must provide a SOW. This includes SOW topics and views, queues, and conflated topics. These features of AMPS are discussed in more detail in the *User Guide*.

Common Options - spark sow

Table 5.2. Spark sow options

Option	Definition
`server`	AMPS server to connect to.
`topic`	Topic to query.
batchsize	Batch Size to use during query. A batch size > 1 can help improve performance, as described in the chapter of the *User Guide* discussing the SOW.
filter	The content filter to use.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.
orderby	An expression that AMPS will use to order the results.
topn	Request AMPS to limit the query response to the first N records returned.
type	For protocols and transports that accept multiple message types on a given transport, specifies the message type to use.

Examples

```
%> ./spark sow -server localhost:9007 -type json -topic order \
```

```
   -filter "/id = '1'"

{ "id" : 1, "data" : "hello, world" }
Total messages received: 1 (Infinity/s)
```

Example 5.6. spark SOW query

This `sow` command will query the *order* topic and filter results which match the xpath expression `/id = '1'`. This query will return the result published in Example 5.3.

If the topic does not provide a SOW, the command returns an error indicating that the command is not valid for that topic.

subscribe

The `subscribe` command allows a `spark` client to query all incoming messages to a topic in real time. Similar to the `sow` command, the `subscribe` command supports regular expression topic matching and content filtering, which allow a query to be very specific when looking for data as it is published to AMPS. Unlike the `sow` command, a subscription can be placed on a topic which does not have a persistent SOW cache configured. This allows a subscribe command to be very flexible in the messages it can be configured to receive.

Common Options - spark subscribe

Table 5.3. Spark subscribe options

Option	Definition
`server`	AMPS server to connect to.
`topic`	Topic to subscribe to.
delta	Use delta subscription (sends a `delta_subscribe` command to AMPS).
filter	Content filter to use.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.
ack	Enable acknowledgements when receiving from a queue. Notice that, when this option is provided, `spark` acknowledges messages from the queue, signalling to AMPS that the message has been fully processed. (See the *User Guide* chapter on AMPS message queues for more information.)
backlog	Request a `max_backlog` of greater than 1 when receiving from a queue. (See the *User Guide* chapter on AMPS message queues for more information.)
type	For protocols and transports that accept multiple message types on a given transport, specifies the message type to use.

Examples

```
%> ./spark subscribe -server localhost:9007 -topic chaos \
                       -type json -filter "/name = 'cup'"

{ "name" : "cup", "place" : "cupboard" }
```

Example 5.7. Spark subscribe example

Example 5.7 places a subscription on the *chaos* topic with a filter that will only return results for messages where `/name = 'cup'`. If we place this subscription before the `publish` command in Example 5.5 is executed, then we will get the results listed above.

sow_and_subscribe

The `sow_and_subscribe` command is a combination of the `sow` command and the `subscribe` command. When a `sow_and_subscribe` is requested, AMPS will first return all messages which match the query and are stored in the SOW. Once this has completed, all messages which match the subscription query will then be sent to the client.

The `sow_and_subscribe` is a powerful tool to use when it is necessary to examine both the contents of the SOW, and the live subscription stream.

Common Options - spark sow_and_subscribe

Table 5.4. Spark sow_and_subscribe options

Option	Definition
`server`	AMPS server to connect to.
`topic`	Topic to query and subscribe to.
batchsize	Batch Size to use during query.
delta	Request delta for subscriptions (sends a `sow_and_delta_subscribe` command to AMPS)
filter	Content filter to use.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.
orderby	An expression that AMPS will use to order the SOW query results.
topn	Request AMPS to limit the SOW query results to the first N records returned.
type	For protocols and transports that accept multiple message types on a given transport, specifies the message type to use.

Examples

```
%> ./spark sow_and_subscribe -server localhost:9007 -type json \
                               -topic chaos -filter "/name = 'cup'"

{ "name" : "cup", "place" : "cupboard" }
```

Example 5.8. spark SOW and subscribe example

In Example 5.8 the same topic and filter are being used as in the `subscribe` example in Example 5.7. The results of this query initially are similar also, since only the messages which are stored in the SOW are returned. If a publisher were started that published data to the topic that matched the content filter, then those messages would then be printed out to the screen in the same manner as a `subscription`.

sow_delete

The `sow_delete` command is used to remove records from the SOW topic in AMPS. If a filter is specified, only messages which match the filter will be removed. If a file is provided, the command reads messages from the file and sends those messages to AMPS. AMPS will delete the matching messages from the SOW. If no filter or file is specified, the command reads messages from standard input (one per line) and sends those messages to AMPS for deletion.

It can be useful to test a filter by first using the desired filter in a `sow` command and make sure the recored returned match what is expected. If that is successful, then it is safe to use the filter for a `sow_delete`. Once records are deleted from the SOW, they are not recoverable.

Common Options - sow_delete

Table 5.5. Spark sow_delete options

Option	Definition
`server`	AMPS server to connect to.
`topic`	Topic to delete records from.
filter	Content filter to use. Notice that a filter of `1=1` is true for every message, and will delete the entire set of records in the SOW.
file	File from which to read messages to be deleted.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.
type	For protocols and transports that accept multiple message types on a given transport, specifies the message type to use.

Examples

```
%> ./spark sow_delete -server localhost:9007 \
   -topic order -type json -filter "/name = 'cup'"

   Deleted 1 records in 10ms.
```

Example 5.9. spark SOW delete example

With the `spark` command in Example 5.9, we are asking for AMPS to delete records in the topic *order* which match the filter `/name = 'cup'`. In this example, we delete the record we published and queried previously

in the `publish` and `sow` spark examples, respectively. `spark` reports that one matching message was removed from the SOW topic.

ping

The spark `ping` command is used to connect to the amps instance and attempt to logon. This tool is useful to determine if an AMPS instance is running and responsive.

Common Options - spark ping

Table 5.6. Spark ping options

Option	Definition
`server`	AMPS server to connect to.
proto	Protocol type to use. In this release, `spark` supports `amps`, `fix`, `nvfix`, `json` and `xml`. Defaults to `amps`.

Examples

```
%> ./spark ping -server localhost:9007 -type json
Successfully connected to tcp://user@localhost:9007/amps/json
```

Example 5.10. Successful ping using spark

In Example 5.10, spark was able to successfully log onto the AMPS instance that was located on port `9007`.

```
%> ./spark ping -server localhost:9119
Unable to connect to AMPS
(com.crankuptheamps.client.exception.ConnectionRefusedException: Unable to
connect to AMPS at localhost:9119).
```

Example 5.11. Unsuccessful ping using spark

In Example 5.11, spark was not able to successfully log onto the AMPS instance that was located on port `9119`. The error shows the exception thrown by spark, which in this case was a `ConnectionRefusedException` from Java.

5.3. Spark Authentication

Spark includes a way to authenticate credentials for development. For example, to subscribe to a a specific user ID and password, simply provide them in the URI in the format `user:password@host:port`.

The command below shows how to use spark to subscribe to a server, providing the specified username and password to AMPS.

```
$AMPS_HOME/bin/spark subscribe -type json \
                               -server username:password@localhost:9007
```

AMPS provides the ability to implement custom authentication. Spark includes a default authentication scheme, which uses the username and password in the -server parameter as described above.

Authentication schemes for `spark` are implemented in Java, using the same interface that the AMPS Java client uses. To use a different authentication scheme with `spark`, you implement the `AuthenticatorFactory` interface in `spark` to return your custom authenticator, adjust the CLASSPATH to include the .jar file that contains the authenticator, and then provide the name of your `AuthenticatorFactory` on the command line. See the *AMPS Java Client* API documentation for details on implementing a custom `Authenticator`.

The command below explicitly loads the default factory, found in the `spark` package, without adjusting the CLASSPATH.

```
$AMPS_HOME/bin/spark subscribe -server username:password@localhost:9007 \
                               -type json -topic foo \
     -authenticator com.crankuptheamps.spark.DefaultAuthenticatorFactory
```

Chapter 6. amps_upgrade

New verisons of AMPS periodically change the format of the data files AMPS uses. The `amps_upgrade` utilty upgrades datafiles from previous versions of AMPS to the current version. `amps_upgrade` supports upgrades from version 3.0.3 or later instances.

6.1. Options and Parameters

Table 6.1. Options for `amps_upgrade`

Option	Description
`--verbose`	Print additional details on each operation to stdout
`--trace`	Print the operations that `amps_upgrade` performs to stdout.

Table 6.2. Instance parameters for `amps_upgrade`

Option	Description
`--from=BASE`	The root directory of the AMPS installation being migrated. This is the directory in which you usually start the `ampServer` process. Any relative paths in the config file will be evaluated relative to this directory.
`--config=CONFIG`	The xml configuration file for the AMPS server being migrated.
`--work-dir=WORK_DIR`	The working directory from which the `ampServer` is invoked.
`--tmp-dir=TMP_DIR`	The temporary directory where upgrade files are written while the upgrade process is underway. If this directory does not exist, it will be created. `amps_upgrade` will fail without changing any existing files if this directory already exists and contains files from a previous migration.

Table 6.3. Actions for `amps_upgrade`

Option	Description
`--check-current`	Returns true if the instance is the same version as the `amps_migrate` utility, with no upgrade needed
`--dry-run`	Step through the entire upgrade process, printing activity, without making changes. Returns `false` if errors are encountered or upgrade is impossible.
`--upgrade`	Upgrade the instance, returning false if the upgrade is impossible or the upgrade process fails.
`-h, --help`	Show usage information and exit.
`--version`	Show the program's version number and exit.

6.2. Usage

Simply upgrade an AMPS instance that's executed in the /amps/server directory from another version to this version, storing temporary files in the /amps/tmp directory:

```
$ amps_upgrade --from=/amps --config=/amps/config.xml --work-dir=/amps/
server --tmp-dir=/amps/tmp --upgrade
```

Example 6.1. upgrading an instance with amps_upgrade

Try out a migration without actually committing the changes to your AMPS instance:

```
$ amps_upgrade --from=/amps --config=/amps/config.xml --work-dir=/amps/
server --tmp-dir=/amps/tmp --dry-run
```

Example 6.2. amps_upgrade dry run

Check to see if your AMPS instance is current:

```
$ amps_upgrade --from=/amps --config=/amps/config.xml --work-dir=/amps/
server --tmp-dir=/amps/tmp --check-current
```

Example 6.3. check to see if an upgrade is needed

Chapter 7. amps-sqlite3

AMPS stores statistics in a sqlite3 database. The amps-sqlite3 script is a convenience wrapper to make it easier to query the AMPS statistics database.

The wrapper provides two main functions:

1. When run, the wrapper creates a set of temporary tables that join the dynamic and static statistics from the database. For example, CPU information for the host system is captured in the HCPUS_STATIC and HCPUS_DYNAMIC tables. For ease of querying, the wrapper joins these into a single HCPU table.

2. The wrapper includes a set of convenience functions for working with date and time. These wrapper functions convert back and forth from the timestamps used in the database to the ISO8601 format used in AMPS logs.

See the *AMPS Monitoring Reference* for details on the statistics captured.

7.1. Parameters

Table 7.1. Parameters for `amps-sqlite3`

Parameter	Description
`database`	The sqlite3 database file to query.
`query`	The query to run. Notice that the query must be enclosed in quotes.

The `amps-sqlite3` wrapper provides a set of convienience functions that can be included in the query. These functions are evaluated before the query is presentd to the sqlite3 database engine.

Table 7.2. Convenience functions in `amps-sqlite3`

Option	Description
`iso8601(`*`timestamp`*`)`	Convert *`timestamp`* to an ISO8601 format string.
`iso8601_local(`*`timestamp`*`)`	Convert *`timestamp`* to an ISO8601 format string in the local timezone.
`timestamp(`*`string`*`)`	Convert the provided ISO8601 format *`string`* to a timestamp.

7.2. Usage

Simply provide the file name of the database to query and the query to run:

```
$ amps-sqlite3 stats.db "select iso8601(timestamp),system_percent from hcpus
 order by timestamp"
```

Example 7.1. returning a histogram of CPU load for the host

Chapter 8. amps_file

AMPS contains a utility that identifies the file type and version number of AMPS files.

8.1. Options and Parameters

Table 8.1. Parameters for `amps_file`

Option	Description
`file_name`	The file name to report on. This argument supports UNIX shell globbing.

8.2. Usage

The following example shows the output of running `amps_file` on a SOW file:

```
%> ./amps_file /amps_dir/sow/mytopic.sow

mytopic.sow: AMPS sow 4.0
```

Example 8.1. Example of ampserr Usage

In this case, the file is recognized as an AMPS SOW file that uses the version 4.0 of the AMPS SOW file format.