AMPS Command Reference Guide

60East Technologies

5.0

Jan 19, 2017

1. AMPS Command Reference Guide

This guide includes a listing of all AMPS commands as well as the required and optional parameters. AMPS supports a consistent set of commands and options regardless of the protocol used to communicate with AMPS. This guide covers the semantics of the commands and options, but does not cover how those commands and options are represented in any particular protocol. Each protocol uses a different concrete format for messages, and that format is specific to the protocol.

To use a command from your application, set the properties of the Command object as shown in this guide, the use the execute function to send the command. The AMPS client is responsible for interpreting the command and formatting the message to AMPS in the proper format for the specific protocol the client is using.

2. Commands

Logging On

logon

Description

To help identify clients and users, it is recommended that clients send a `logon` command to the AMPS engine and specify a client name.

AMPS only allows a single `logon` command for each connection. The `logon` command must be the first command sent over a new connection. Otherwise, AMPS performs an implicit `logon`, causing any other `logon` commands for the connection to be rejected.

In AMPS configurations where authentication is enabled, all connecting clients must issue a `logon` message with the `username` and `password` credentials specified in the command. Attempts to logon to an AMPS instance that do not contain the information required will be rejected and prohibited from issuing further commands until a successful `logon` has been placed.

If an AMPS client is connected to an instance that has journaling enabled, the `ClientName` specified *must* be unique - otherwise, the `logon` will fail. In the failure case, the acknowledgement message returned will contain a `Status` of 'failure' and a `Reason` of 'name in use'.

It is recommended that all `logon` commands request that a `processed` acknowledgement message be requested in the `AckType` header of the `logon` message. This will allow AMPS to communicate the result of the `logon` command to the client, allowing the client to determine how to best proceed.

Header Fields

Table 1 contains the header fields available to a `logon` command.

Table 1. Header fields supported by `logon`

Field	Description
`Command`	The command to be executed. Value: `logon`.
`ClientName`	A string identifier used to give a client a unique id. AMPS does not limit the character set used in this name. However, the specific protocol may have character set limitations. 60East recommends that the client name is meaningful, short, human readable, and avoids using control characters, newline characters, or square brackets.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received` or `processed` .
`SequenceId`	The sequence ID of the last message received by the client. Passing in the sequence ID of the last processed bookmark will cause AMPS to replay the transaction log from the bookmark up to the most recent message persisted in the transaction log.
`UserId`	The username passed into the AMPS authentication and entitlement module.
`Password`	The password passed into the AMPS authentication and entitlement module.
`CorrelationId`	A user-provided string that will be included in the log message recording this logon, and in the information provided for the connection in the administration interface. AMPS does not interpret this string or use the string for any other purpose. If this header is not present, AMPS does not store a value for the `CorrelationId` for this connection. The contents of this header must consist of characters that are legal in Base64 encoding.

Returns

A `logon` message specifying an `AckType` of `received` or `processed` will receive an `ack` message to acknowledge the message receipt. If a client requests an acknowledgment message, the header will also contain the `ClientName` which was part of the original `logon` message.

When requested, the `logon` command will result in a `processed` acknowledgment message. This returned acknowledgment is used in determining if a client was successfully authenticated against a server which has an authentication module enabled.

Table 2 contains the acknowledgment messages that can be returned by a `logon` command.

Table 2. Acknowledgment messages supported by `logon`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	Not supported at this time.
`persisted`	Not supported at this time.
`processed`	AMPS has processed the `logon` message.
`received`	The `logon` command has been received.
`stats`	Not supported at this time.

Publishing to AMPS

delta_publish

Description

The `delta_publish` command is a way of publishing an incremental update to a record. If a client uses `delta_publish` to publish an update, AMPS first extracts the key fields from the record and does a look up for the record in the SOW. AMPS will then apply the update to the SOW record overwriting any field that has a newer value in the update and appending to the record any new fields that were not previously in the SOW record.

If `delta_publish` is used on a record that does not currently exist in the SOW or if it is used on a topic that does not have a SOW-topic store defined, then `delta_publish` will behave like a standard `publish` command.

A `delta_publish` is transparent to other clients and the merged record will be forwarded to matching subscriptions.

Header Fields

Table 3 contains the header fields available to a `delta_publish` command.

Table 3. Header fields used in a `delta_publish`

Field	Description
`Command`	Command to be executed. Value: `delta_publish`
`Topic`	The SOW topic to publish the message to.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received`, `processed`, `completed` or `stats`.
`CommandId`	If specified with an AMPS command which requests an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the `ack` response header.

Field	Description
`Expiration`	An interval used to define the lifetime of a `delta_message` message. Time period is in seconds.
`Sequence`	A monotonically increasing number used to identify published messages in a high availability environment.
`TransmissionTime`	An ISO-8601 datetime used to not the time the message is sent by the client.
`CorrelationId`	A user-provided string that will be passed, verbatim, to subscribers. If this header is not present, subscribers receive no value for the `CorrelationId`. The contents of this header must consist of characters that are legal in Base64 encoding.
`SowKey`	For SOW topics that use an explicit key, the SOW key to use for the message. The contents of this header must consist of characters that are legal in Base64 encoding.

Returns

For a `delta_publish` message, AMPS will send acknowledgment messages for the following `AckType` fields: `received`, `processed` and `persisted` along with a populated `Status` header field describing the acknowledgment.

Table 4 contains the acknowledgment messages that can be returned by a `delta_publish`.

Table 4. Acknowledgment messages supported by `delta_publish`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	Not supported at this time.
`persisted`	When AMPS returns an acknowledgment message of `persisted`, it guarantees that: 1. All downstream synchronous replications have acknowledged that the message(s) have been delivered to their respective SOW Topic(s). 2. When the publish message has been sent to all available downstream asynchronous replications.
`processed`	AMPS has processed the message(s) to be published to the SOW. Any errors which occur in the message will be returned to the client in this acknowledgement message.
`received`	The `delta_publish` message has been received.
`stats`	Not supported at this time.

Errors

Any errors that occur during this command will be returned in the status of a `processed` acknowledgement message and logged to the log file. Regardless of success or failure, the `processed` acknowledgment message will be returned only if requested by specifying `processed` in the `AckType` field.

publish

Description

The `publish` command is the primary way to inject messages into the AMPS processing stream. A `publish` command received by AMPS will be forwarded to other connected clients with matching subscriptions.

Header Fields

Table 5 contains the header fields available to a `publish` command.

Table 5. Header fields supported by `publish`

Field	Description
`Command`	Command to be executed. Value: `publish`.
`Topic`	The topic to publish the message to.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one of more of the following: `none`, `received`, `persisted` or `processed`.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message in response to the `publish` command, all requested acknowledgment messages will contain the `CommandId` in the response header.
`Expiration`	An interval in seconds, used to define the lifetime of a `publish` message.
`SequenceId`	A monotonically increasing identifier used in high availability configurations to determine message uniqueness across replicas.
`TransmissionTime`	An ISO-8601 datetime used to note the time the message is sent by the client.
`CorrelationId`	A user-provided string that will be passed, verbatim, to subscribers. If this header is not present, subscribers receive no value for the `CorrelationId`. The contents of this header must consist of characters that are legal in Base64 encoding.
`SowKey`	For SOW topics that use an explicit key, the SOW key to use for the message. The contents of this header must consist of characters that are legal in Base64 encoding.

Returns

A client which issues a `publish` can request a `processed` acknowledgment message; however this is not recommended as there is a significant performance overhead associated with this. Table 6 contains the `AckType` messages which can be returned by a `publish`.

Table 6. Acknowledgment messages supported by `publish`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	Not supported at this time.
`persisted`	When AMPS returns an acknowledgment message of `persisted`, it guarantees that:

Acknowledgment	Description
	1. All downstream synchronous replications have acknowledged that the message(s) have been delivered to their respective SOW Topic(s). 2. When the `publish` message has been sent to all available downstream asynchronous replications.
`processed`	AMPS has processed the `publish` message.
`received`	The `publish` message has been received.
`stats`	Not supported at this time.

Errors

Any errors that occur during this command will be returned in the status of a `processed` acknowledgment and logged to the log file. Regardless of success or failure, the `processed` acknowledgment will be returned only if the request includes the `processed` in the `AckType` field.

Querying and Subscribing

delta_subscribe

Description

The `delta_subscribe` command is like the `subscribe` command except that subscriptions placed through `delta_subscribe` will receive only messages that have changed between the SOW record and the new update.

If `delta_subscribe` is used on a record which does not currently exist in the SOW or if it is used on a topic which does not have a SOW-topic store defined, then `delta_subscribe` behaves like a `subscribe` command.

Header Fields

Table 7 contains the header fields available to a `delta_subscribe` command.

Table 7. Header fields supported by `delta_subscribe`

Field	Description
`Command`	Command to be executed. Value: `delta_publish`
`Topic`	Topic with which to place a subscription.
`AckType`	Acknowledgment type for the `delta_subscribe` command. Value is a comma separated list of one or more of the following: `none`, `received`, `processed`, `completed` or `stats`.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the acknowledgment response header.

Field	Description
`DataOnly`	A Boolean (`true` or `false`) used to determine the type of data sent to the subscriber. A value of `true` will, for example, not include a SOAP envelope.
`Filter`	String which is used as a content filter expression. When using XML, the filter must be wrapped in a `CDATA`.
`Options`	A comma separated list of flags available to the `subscribe` command. Table 8 describes the `Options` available for use in the `delta-subscribe` command.
`SendEmpty`	Boolean (`true` or `false`) value used to determine whether empty messages which are published will be forwarded to matching subscriptions. The default value is `true`.
`SendSubscriptionIds`	Boolean (`true` or `false`) subscription identifiers will not be sent for all matched messages if set to `false`.
`SubscriptionId`	The subscription ID for this command. When provided with a new subscription, this is the identifier that AMPS will use for the subscription. When provided with the `replace` option, this field specifies the subscription to replace. When provided with a `pause` or `resume` option, this field specifies the subscriptions to pause or resume. For a new subscription, the AMPS clients will generate a subscription ID if one is not provided.
`TransmissionTime`	An ISO-8601 datetime used to note the time the message is sent by the client.
`Bookmark`	A bookmark specifying the point in the transaction log at which to start the subscription. If the topic provided is not recorded in a transaction log, AMPS enters the subscription without replaying messages. You can provide a single bookmark, or a comma-delimited list of bookmarks. When a list is provided, AMPS starts the subscription at the earliest bookmark in the list.

Options Field

Table 8 contains a list of the `Options` available and their definitions when used in the AMPS `sow_and_delta_subscribe` command.

Table 8. `Options` types supported by `delta_subscribe`

Option	Description
`none`	This is the default `Options` type.
`live`	Tells AMPS to send messages to subscribing clients before they have been persisted to the transaction log. This option is only valid for bookmark subscriptions.
`max_backlog=`*n*	When subscribing to a queue, the number of unacknowledged messages the client is willing to accept at a time. AMPS will not exceed this number, but may choose a smaller number depending on the queue configuration.
`no_empties`	Tells AMPS not to send empty publish messages to matching subscriptions. This can be useful for suppressing messages where no fields have changed.
`no_sowkeys`	Tells AMPS not to send the AMPS-generated SowKey for messages.

Option	Description
`oof`	Send an OOF message for records which have fallen out of focus from the original subscription.
`pause`	Pause a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS pauses the subscription or subscriptions provided in the `SubId` of the command.
`rate=`*n*	Set the maximum message delivery rate for a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. The rate can be specified as either the number of messages per second (for example, `1000`) or the number of bytes per second (for example, `100KB`).
`replace`	Replace the subscription associated with `CmdId` with another subscription. When provided as part of `sow_and_subscribe`, AMPS runs a SOW query for the new subscription.
`resume`	Resume a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS resumes the subscription or subscriptions provided in the `SubId` of the command.
`send_keys`	AMPS will send the SOW keys back with matching mesages from the SOW.
`timestamp`	AMPS will include a header with the time at which AMPS processed the message.

Returns

For a `delta_subscribe` message, AMPS will send acknowledgment messages for the following `AckType` fields: `received`, `processed`, `persisted` and `stats` along with a populated `Status` header field describing the acknowledgment.

Table 9 contains the `AckType` messages which can be returned by a `delta_subscribe`.

Table 9. Acknowledgment messages supported by `delta_publish`

Acknowledgment	Description
`none`	No `ack` is returned. This is the default behavior.
`completed`	When a bookmark is present on the subscribe request and this acknowledgement is requested, AMPS sends a `completed` acknowledgment message to indicate that bookmark replay is complete. Further messages on this subscription are from new publishes.
`persisted`	When a bookmark is present and this acknowledgement is requested, AMPS periodically sends a `persisted` acknowledgement message to indicate the most recent bookmark in the server's transaction log.
`processed`	AMPS has compiled the filters for the `delta_subscribe` message(s).
`received`	The `delta_subscribe` message has been received.
`stats`	Returns an acknowledgment message with `Matches`, `TopicMatches` and `RecordsReturned`.

Errors

Any errors that occur during this command will be returned in the status of a `processed` acknowledgment and logged to the log file. Regardless of success or failure, the `processed` acknowledgment will be returned only if requested by including processed in the `AckType` field of the `delta_subscribe` message header.

sow_and_delta_subscribe

Description

A `sow_and_delta_subscribe` command is used to combine the functionality of commands `sow` and a `delta_subscribe` in a single command.

The `sow_and_delta_subscribe` command is used (a) to query the contents of a SOW topic (this is the `sow` command); and (b) to place a subscription such that any messages matching the subscribed SOW topic and query filter will be published to the AMPS client (this is the `delta_subscribe` command). As with the `delta_subscribe` command, `publish` messages representing updates to SOW records will contain only the information that has changed.

If a `sow_and_delta_subscribe` is issued on a record that does not currently exist in the SOW topic, or if it is used on a topic that does not have a SOW-topic store defined, then a `sow_and_delta_subscribe` will behave like a `sow_and_subscribe` command.

Header Fields

Table 10 contains the header fields supported by a `sow_and_delta_subscribe` command.

Table 10. Header fields supported by sow_and_delta_subscribe

Field	Description
`Command`	The command to be executed. Value: `sow_and_delta_subscribe`
`Topic`	The target SOW topic to query and subscribe to.
`AckType`	Acknowledgment type for the given command. Value is a comma separated string of one of more of the following: `none`, `received`, `processed`, `completed` or `stats`.
`BatchSize`	Number of records to return in a single sow query results message. While the default value is 1, it is recommended to use a higher value, as even small increases can yield greater performance in query result delivery.
`Bookmark`	A bookmark specifying the historical state of the SOW to return results from. For SOW topics where historical query is enabled, AMPS returns the saved state of the SOW as of that bookmark. For SOW topics where historical query is not enabled, AMPS ignores this parameter. If the topic is enabled for historical query and AMPS has a transaction log that covers the topic, AMPS returns the saved state of the SOW as of that bookmark and starts a bookmark subscription at a point in the transaction log immediately after the point at which the SOW state was saved. In other words, if the granularity of the historical SOW preserves the state of the SOW at 11:30:10 AM and 11:30:50 AM, a request for a bookmark at 11:30:20

Field	Description
	AM will provide the SOW state as of 11:30:10 AM, and begin the replay immediately after that SOW state. This ensures no messages are missed, but means that the subscription may begin before the bookmark.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message, all `ack` messages will contain the `CommandId` in the acknowledgment message.
`DataOnly`	If `true`, send only raw data to subscriber for a matching publish message. For example: this will remove the SOAP envelope in an XML message.
`Filter`	Content filter expression.
`Options`	A comma separated list of one or more of the following: `none`, `live`, `no_empties`, `oof`, `replace` or `send_keys`. Table 11 describes all of the `Options` available.
`OrderBy`	Return the SOW results sorted by the specified fields. Fields are a comma-delimited list of AMPS identifiers, and may optionally include a sort specifer, `ASC` or `DESC`.
`QueryId`	Identifier used to identify the client's SOW topic query. This identifier will be added to all messages that represent a response to the `sow_and_delta_subscribe` command.
`SendEmpty`	If set to `true`, empty published messages are forwarded to matching subscriptions. Default is `true`.
`SendOOF`	Messages that have fallen out of focus from the subscription are sent to the client. Default is `false`.
`SendKeys`	Option to instruct AMPS that the client would like to receive the `SowKey` back.
`SendSubscriptionIds`	If `true` subscription identifiers will be sent for a matched message.
`SowKeys`	A comma-delimited list of `SowKeys` that identify the messages to return from the query. For example, you might send a query with the SowKeys value `42,100,3467` which would return records with those SowKey values, if any exist in the SOW.
`SubscriptionId`	The subscription ID for this command. When provided with a new subscription, this is the identifier that AMPS will use for the subscription. When provided with the `replace` option, this field specifies the subscription to replace. When provided with a `pause` or `resume` option, this field specifies the subscriptions to pause or resume. For a new subscription, the AMPS clients will generate a subscription ID if one is not provided.
`TopN`	Return up to the number of messages specified from the SOW query.
`TransmissionTime`	An ISO-8601 datetime used to note the time the message is sent by the client.

Options Field

Table 11 contains a list of the `Options` available and their definitions when used in the AMPS `sow_and_delta_subscribe` command.

Table 11. Options supported by `sow_and_delta_subscribe`

Option	Description
`none`	This is the default `Options` type.
`live`	Tells AMPS to send messages to subscribing clients before they have been persisted to the transaction log.
`no_empties`	Tells AMPS not to send empty publish messages to matching subscriptions. This can be useful for suppressing messages where no fields have changed.
`no_sowkeys`	Tells AMPS not to send the AMPS-generated `SowKey` for messages.
`oof`	Send an `OOF` message for records that have fallen out of focus from the original subscription.
`replace`	Replace the subscription associated with `CmdId` with another subscription.
`send_keys`	AMPS will send the SOW keys (that is, the data fields used to identify unique messages in the SOW) back with matching messages from the SOW.
`timestamp`	AMPS will include a header with the time at which AMPS processed the message.

Returns

AMPS will send acknowledgment messages for the following `AckType` fields: `received` and `processed`, along with a populated `Status` header field describing the acknowledgment message.

If the `sow_and_delta_subscribe` command is successful, AMPS will return a `group_begin` message to notify the client that a group of messages is being returned as part of the `sow` portion of the command. The *SOW Queries* chapter in the *AMPS User Guide*provides more information about SOW topic query behavior. Table 12 contains the `AckType` messages which can be returned by a `sow_and_delta_subscribe`.

Table 12. Acknowledgment messages supported by `sow_and_delta_subscribe`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	The `sow_and_delta_subscribe` message has completed the `sow` portion of the command, and all future messages will be updated based on publishes.
`persisted`	Not supported at this time.
`processed`	AMPS has compiled the filter(s) for the `sow_and_delta_subscribe` message(s).
`received`	The `sow_and_delta_subscribe` message has been received.
`stats`	Returns an `ack` message with `Matches`, `TopicMatches` and `RecordsReturned`.

The `stats` acknowledgment message includes three values in the header, the `Matches`, `TopicMatches` and the `RecordsReturned`. These are defined below:

TopicMatches. The total number of records compared across all matching SOW topics.

Matches . The number of records returned that match the topic regular expression and the content filter. This value can be greater than `RecordsReturned` in the case where the number of returned records is limited by `TopN`.

RecordsReturned . The total number of records returned to the client, which can be limited by the `TopN` header value.

Errors

Errors for a `sow_and_delta_subscribe` query are either returned in the `Status` field if an `AckType` has been defined, or the errors may be inserted into the AMPS log.

sow_and_subscribe

Description

A `sow_and_subscribe` command is used to combine the functionality of `sow` and a `subscribe` command in a single command.

The `sow_and_subscribe` command is used (a) to query the contents of a SOW topic (this is the `sow` command); and (b) to place a subscription such that any messages matching the subscribed SOW topic and query filter will be published to the AMPS client (this is the `subscribe` command). As with the `subscribe` command, `publish` messages representing updates to SOW records will contain only information that has changed.

Header Fields

Table 13 contains the header fields supported by a `sow_and_subscribe` command.

Table 13. Header fields supported by `sow_and_subscribe`

Field	Description
`Command`	The command to be executed. Value: `sow_and_subscribe`.
`Topic`	The target SOW topic to query and subscribe.
`AckType`	Acknowledgment type for the given command. Value is a comma separated string of one of more of the following: `none`, `received`, `processed`, `completed` or `stats`.
`BatchSize`	Number of records to return in a single sow query results message. While the default value is 1, it is recommended to use a higher value, as even small increases can yield greater performance in query result delivery.
`Bookmark`	A bookmark specifying the historical state of the SOW to return results from. For SOW topics where historical query is enabled, AMPS returns the saved state of the SOW as of that bookmark. For SOW topics where historical query is not enabled, AMPS ignores this parameter. If the topic is enabled for historical query and AMPS has a transaction log that covers the topic, AMPS returns the saved state of the SOW as of that bookmark and starts a bookmark subscription at a point in the transaction log immediately after the point at which the SOW state was saved. In other words, if the granularity of the historical SOW preserves the state of the SOW at 11:30:10 AM and 11:30:50 AM, a request for a bookmark at 11:30:20

Field	Description
	AM will provide the SOW state as of 11:30:10 AM, and begin the replay immediately after that SOW state. This ensures no messages are missed, but means that the subscription may begin before the bookmark.
`CommandId`	If specified with an AMPS command requesting an acknowledgement message, all `ack` messages will contain the `CommandId` in the acknowledgment message.
`DataOnly`	Only send raw data to subscriber for a matching `publish` message if `true`. For example: this will remove the SOAP envelop in an XML message.
`Filter`	Content filter expression.
`Options`	A comma separated list of flags available to the `sow_and_subscribe` command. Table 15 describes the options available.
`OrderBy`	Return the SOW results sorted by the specified fields. Fields are a comma-delimited list of AMPS identifiers, and may optionally include a sort specifer, `ASC` or `DESC`.
`QueryId`	Identifier used to identify the client's SOW topic query. This identifier will be added to all messages representing a response to the `sow_and_subscribe` command.
`SendOOF`	Messages that have fallen out of focus from the subscription are sent to the client. Default is `false`.
`SendKeys`	Option to instruct AMPS that the client would like to receive the `SowKey` back.
`SendSubscriptionIds`	If `true`, subscription identifiers will be sent for a matched message.
`SowKeys`	A comma-delimited list of `SowKeys` that identify the messages to return from the query.
`SubscriptionId`	The subscription ID for this command. When provided with a new subscription, this is the identifier that AMPS will use for the subscription. When provided with the `replace` option, this field specifies the subscription to replace. When provided with a `pause` or `resume` option, this field specifies the subscriptions to pause or resume. For a new subscription, the AMPS clients will generate a subscription ID if one is not provided.
`TopN`	Return up to the number of messages specified from the SOW query.
`TransmissionTime`	An ISO-8601 datetime used to note the time the message is sent by the client.

Returns

AMPS will send acknowledgment messages for the following `AckType` fields: `received`, `processed` along with a populated `Status` header field describing the acknowledgment message.

If the `sow_and_subscribe` command is successful, AMPS will return a `group_begin` message to notify the client that a group of messages is being returned as part of the `sow` portion of the command.

The *SOW Queries* chapter in the *AMPS User Guide* will provide more information about SOW topic query behavior. Table 14 contains the `AckType` messages that can be returned by a `sow_and_subscribe`.

Table 14. Acknowledgment messages supported by `sow_and_subscribe`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	The `sow_and_subscribe` message has completed the `sow` portion of the command, and all future messages will be updated based on publishes.
`persisted`	Not supported at this time.
`processed`	AMPS has completed the work necessary to register the subscription and begin the SOW query.
`received`	The `sow_and_subscribe` message has been received.
`stats`	Returns an `ack` message with `Matches`, `TopicMatches` and `RecordsReturned`.

The `stats` acknowledgment message includes three values in the header, the `Matches`, `TopicMatches` and the `RecordsReturned`. These are defined below:

TopicMatches. The total number of records compared across all matching SOW topics.

Matches . The number of records returned that match the topic regular expression and the content filter. This value can be greater than `RecordsReturned` in the case where the number of returned records is limited by `TopN`.

RecordsReturned . The total number of records returned to the client, which can be limited by the `TopN` header value.

Options Field

Table 15 contains a list of the `Options` available and their definitions when used in the AMPS `sow_and_subscribe` command.

Table 15. Options types supported by `sow_and_subscribe`

Option	Description
`none`	This is the default `Options` type.
`live`	Tells AMPS to send messages to subscribing clients before they have been persisted to the transaction log. This option is only valid for bookmark subscriptions.
`no_sowkeys`	Tells AMPS not to send the AMPS-generated `SowKey` for messages.
`oof`	Send on `OOF` message for records which have fallen out of focus from the original subscription.
`pause`	Pause a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS pauses the subscription or subscriptions provided in the `SubId` of the command.
`rate=`*n*	Set the maximum message delivery rate for a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. The rate can be specified as either the number of messages per second (for example, `1000`) or the number of bytes per second (for example, `100KB`).

Option	Description
`replace`	Replace the subscription associated with `CmdId` with another subscription. When provided as part of `sow_and_subscribe`, AMPS runs a SOW query for the new subscription.
`resume`	Resume a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS resumes the subscription or subscriptions provided in the `SubId` of the command.
`send_keys`	AMPS will send the SOW keys (that is, the data fields used to identify unique messages in the SOW) back with matching messages from the SOW.
`timestamp`	AMPS will include a header with the time at which AMPS processed the message.

Errors

Errors for a `sow_and_subscribe` query are either returned in the `Status` field if an `AckType` has been defined, or the errors may be inserted into the AMPS log.

subscribe

Description

The `subscribe` command is the primary way to retrieve messages from the AMPS processing stream. A client can issue a `subscribe` command on a topic to receive all published messages to that topic in the future. Additionally, content filtering can be used to choose which messages the client is interested in receiving.

Header Fields

Table 16. Header fields supported by `subscribe`

Field	Description
`Command`	Command to be executed. Value: `subscribe`.
`Topic`	Topic to place a subscription against.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received`, `processed` or `completed`.
`Bookmark`	A bookmark specifying the point in the transaction log at which to start the subscription. If the topic provided is not recorded in a transaction log, AMPS enters the subscription without replaying messages. You can provide a single bookmark, or a comma-delimited list of bookmarks. When a list is provided, AMPS starts the subscription at the earliest bookmark in the list.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the `ack` response header.

Field	Description
`DataOnly`	A Boolean value (`true` or `false`) which, if `true`, will send only raw data to subscriber for a matching publish message. In the case where the message type is XML, the SOAP envelope will not be included.
`Filter`	A CDATA wrapped string, used as a content filter expression.
`Options`	A comma separated list of flags available to the `subscribe` command. Table 17 describes the `Options` available for use in the `subscribe` command.
`SendSubscriptionIds`	Boolean (`true` or `false`) that tells if `true` requests AMPS to send subscription identifiers with a matched message.
`SubscriptionId`	The subscription ID for this command. When provided with a new subscription, this is the identifier that AMPS will use for the subscription. When provided with the `replace` option, this field specifies the subscription to replace. When provided with a `pause` or `resume` option, this field specifies the subscriptions to pause or resume. For a new subscription, the AMPS clients will generate a subscription ID if one is not provided.
`TopN`	The maximum number of messages to provide from a bookmark subscription. This parameter is only valid for replay from the transaction log. This parameter is not valid if no bookmark is provided, if the provided bookmark is `0|1|` (start from now), or if the command includes the `live` option.
`TransmissionTime`	An ISO-8601 datetime used to note the time the message is sent by the client.

Options Field

Table 17 contains a list of the `Options` available and their definitions when used in the AMPS `subscribe` command.

Table 17. `Options` types supported by `subscribe`

Option	Description
`none`	This is the default `Options` type.
`live`	Tells AMPS to send messages to subscribing clients before they have been persisted to the transaction log. This option is only valid for bookmark subscriptions.
`max_backlog=`*n*	When subscribing to a queue, the number of unacknowledged messages the client is willing to accept at a time. AMPS will not exceed this number, but may choose a smaller number depending on the queue configuration.
`no_empties`	Not supported by this command type.
`no_sowkeys`	Tells AMPS not to send the AMPS-generated SowKey for messages.
`oof`	Not supported by this command type.
`pause`	Pause a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS pauses the subscription or subscriptions provided in the `SubId` of the command.

Option	Description
`rate=`*n*	Set the maximum message delivery rate for a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. The rate can be specified as either the number of messages per second (for example, `1000`) or the number of bytes per second (for example, `100KB`).
`replace`	Replace the subscription associated with `CmdId` with another subscription.
`resume`	Resume a bookmark subscription. This option is only valid for bookmark subscriptions that do not use the `live` option. When this option is present, AMPS resumes the subscription or subscriptions provided in the `SubId` of the command.
`send_keys`	Not supported by this command type.
`timestamp`	AMPS will include a header with the time at which AMPS processed the message.

Returns

It is possible to specify a `processed` acknowledgment be sent back to the client that issued the `subscribe` command. Within this `processed` acknowledgment, a client can get back the result of placing the subscription (success or failure) and the `SubscriptionId`, which uniquely identifies the subscription within AMPS. Keeping track of the `SubscriptionId` is useful for unsubscribing from subscriptions and issuing SOW queries.

Table 18 contains a list of the supported acknowledgment messages available to the `subscribe` command.

Table 18. Acknowledgment messages supported by `subscribe`

Acknowledgment	*Description*
`none`	No acknowledgment message is returned. This is the default behavior.
`completed`	When a bookmark is present on the subscribe request and this acknowledgement is requested, AMPS sends a `completed` acknowledgment message to indicate that bookmark replay is complete. Further messages on this subscription are from new publishes.
`processed`	AMPS has completed the work necessary to register the subscription. When a bookmark is present and this acknowledgement is requested, this acknowledgement indicates that AMPS is about to begin replay.
`persisted`	When a bookmark is present and this acknowledgement is requested, AMPS periodically sends a `persisted` acknowledgement message to indicate the most recent bookmark in the server's transaction log.
`received`	The `subscribe` message has been received.

Errors

Any errors that occur during this command will be returned in the status of a `processed` acknowledgment and logged to the log file. Regardless of success or failure, the processed acknowledgment will only be returned if requested by including `processed` in the `AckType` field.

sow

Description

The `sow` command is use to query the contents of a previously defined SOW Topic. A `sow` command can be used to query an entire SOW Topic, or a filter can be used to further refine the results found inside a SOW Topic. For more information, see the *State of the World* and *SOW Queries* chapters in the *AMPS User Guide*

Header Fields

Table 19. Header fields supported by `sow`

Field	Description
`Command`	Command to be executed. Value: `sow`.
`Topic`	The SOW topic from which the records will be queried.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received`, `processed`, `completed` or `stats`
`BatchSize`	Number of records to return in a single `sow` query result message. While the default value is 1, it is recommended to use a higher `BatchSize` value, as even small increases can yield greater performance in query result delivery.
`Bookmark`	A bookmark specifying the historical state of the SOW to return results from. For SOW topics where historical query is enabled, AMPS returns the saved state of the SOW as of that bookmark. For SOW topics where historical query is not enabled, AMPS ignores this parameter.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the `ack` response header.
`Filter`	Content filter expression. See the *Content Filtering* chapter in the *AMPS User Guide* for more information on using content filters.
`OrderBy`	Return the SOW results sorted by the specified fields. Fields are a comma-delimited list of AMPS identifiers, and may optionally include a sort specifer, `ASC` or `DESC`.
`QueryId`	Unique identifier which is returned as part of the response delivered back to the client.
`SowKeys`	A comma-delimited list of `SowKeys` that identify the messages to return from the query.
`TopN`	Return up to the number of messages specified from the SOW query.

Options Field

Table 20 contains a list of the `Options` available and their definitions when used in the AMPS `sow` command.

Table 20. Options types supported by `sow`

Option	Description
`none`	This is the default `Options` type.
`no_sowkeys`	Tells AMPS not to send the AMPS-generated `SowKey` for messages.
`oof`	Send on `OOF` message for records which have fallen out of focus from the original subscription.
`replace`	Replace the subscription associated with `CmdId` with another subscription. When provided as part of `sow_and_subscribe`, AMPS runs a SOW query for the new subscription.
`send_keys`	AMPS will send the SOW keys (that is, the data fields used to identify unique messages in the SOW) back with matching messages from the SOW.
`timestamp`	AMPS will include a header with the time at which AMPS processed the message.

Returns

When a `sow` message is received, AMPS can return a `received` message as notification that the message has arrived. When the message filter has been processed, AMPS will return the `processed` acknowledgment message along with any errors that might have occurred.

The results returned by a SOW are put into a `sow` record group by first sending a `group_begin` message, followed by the matching SOW records. A `group_end` message is used to denote the close of query results processing.

Table 21 contains a listing of the acknowledgment messages supported by the `sow` command.

Table 21. Acknowledgment messages returned by `sow`

Acknowledgment	Description
`none`	No acknowledgment message is returned. This is the default behavior.
`completed`	The `sow` command has completed.
`persisted`	Not supported at this time.
`processed`	AMPS has compiled the filter(s) for the `sow` message.
`received`	The `sow` command has been received.
`stats`	Returns statistics related to the state of the SOW query results.

The stats message include three values in the header: `Matches`, `TopicMatches`, and the `RecordsReturned`. These are defined below:

TopicMatches. The total number of records compared across all matching SOW topics.

Matches. The number of records returned that match the topic regular expression and the content filter. This value can be greater than `RecordsReturned` in the case where the number of returned records is limited by `TopN`.

RecordsReturned. The total number of records returned to the client, which can be limited by the `TopN` header value.

Errors

Any errors which occur during a `sow` command are returned in the `processed` acknowledgement message. The error is identified in the `Status` header field in the acknowledgment message, and the reason given in the `Reason` header field.

> ⚠ The ordering of records returned by a SOW query is undefined.

unsubscribe

Description

The `unsubscribe` command allows a client to notify AMPS that it no longer wishes to receive messages related to a previous subscription.

There are two ways that a client can unsubscribe from an existing subscription:

1. Adding the `all` keyword to the `SubId` header field in the `unsubscribe` message will unsubscribe the client from all AMPS SOW topic subscriptions.
2. With each `subscription` command issued, AMPS will return a `SubId` with the `processed` acknowledgment message. Issuing an `unsubscribe` command using the same `SubId` header field which was returned as part of the original `subscribe` command's `processed` acknowledgment message will unsubscribe a client from a single subscription.

Header Fields

Table 22. Header fields supported by `unsubscribe`

Field	Description
`Command`	Command to be executed. Value: `unsubscribe`.
`SubId`	Subscription ID returned from AMPS to the client when the original `subscription` was placed. The keyword `all` can also be used to unsubscribe from all current subscriptions for the client.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received` or `persisted`.
`CommandId`	If specified within an AMPS command requesting an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the `ack` response header.

Returns

The `unsubscribe` command supports the `received` and `processed` acknowledgment message types, as described in Table 23.

Table 23. Acknowledgment messages supported by `unsubscribe`

Acknowledgment	Description
`none`	No acknowledgment message is returned. This is the default behavior.
`completed`	Not supported at this time.
`processed`	AMPS has processed the `unsubscribe` message(s).
`persisted`	Not supported at this time.
`received`	The `unsubscribe` message has been received.
`stats`	Not supported at this time.

Removing Messages

sow_delete

Description

In AMPS, there are three different ways to remove records from the SOW. The first method is to construct a `publish` message that matches the message to be removed, with the `Command` field set to be a `sow_delete` message. This has the net effect of causing AMPS recreate the `SowKey` for the particular message, then look up the `SowKey` message in the SOW and finally remove it.

The other method to remove messages from the SOW is to construct a `sow_delete` message and pass in a comma separated list of `SowKey`s in the message header which will cause all of the messages to be removed from the SOW Topic.

The third way to remove records from the SOW is similar to the manner in which a `sow` query command with a `filter` is performed. In this case, instead of returning the results of the `sow` command, those records that match the filter will be deleted from the SOW.

Header Fields

Table 24 contains the header fields supported by a `sow_delete`.

Table 24. Header fields supported by `sow_delete`

Field	Description
`Command`	Command to be executed. Value: `sow_delete`.
`Topic`	The SOW Topic from which to delete the messages(s).
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `received`, `processed`, `persisted`, `completed` and `stats`.
`CommandId`	If specified with an AMPS command requesting an `ack`, all requested acknowledgment messages will contain the `CommandId` in the acknowledgment message header.

Field	Description
`SowKeys`	A comma separated list of unique ids to be deleted.
`Filter`	Content filter expression. See the *Content Filtering* chapter in the *AMPS User Guide* for more information on using content filters.

The `SowKeys` and `Filter` header fields cannot be used together. They are mutually exclusive. Using them together in the same `sow_delete` command will cause indeterminate results.

Returns

For a `sow_delete` message, AMPS will send acknowledgment message, `completed` and `stats` for the following acknowledgment message types: `received`, `processed` and `persisted` along with a populated `Status` header field describing the acknowledgment.

Table 25. Acknowledgment messages types supported by `sow_delete`

Acknowledgment	Description
`none`	No acknowledgment message is returned. This is the default behavior.
`completed`	Supported for a `sow_delete` with a `Filter` defined. The `completed` acknowledgment message is returned when the query portion of the command has completed.
`persisted`	When an AMPS engine returns an acknowledgment message of `persisted` this guarantees that 1. All downstream synchronous replication(s) have acknowledged that the message(s) have been deleted from their respective SOW topic(s). 2. The `sow_delete` message has been sent to all available downstream asynchronous replications.
`processed`	AMPS has compiled the filter(s) for the `sow_delete` messages.
`received`	The `sow_delete` message has been received.
`stats`	Returns an acknowledgment message with `Matches`, `TopicMatches` and `RecordsDeleted`.

The `stats` acknowledgment message include three values in the header, the `Matches`, `TopicMatches` and the `RecordsDeleted`. These are defined below:

TopicMatches. The total number of records compared across all matching SOW topics.

Matches. The number of records returned that match the topic regular expression and the content filter.

RecordsDeleted . The total number of records deleted.

Errors

Errors that occur during a `sow_delete` are returned as part of the `processed` acknowledgment message and recorded to the log. Typical errors involved a missing topic, or a missing/invalid `SowKey`.

Utility Commands

flush

Description

Sends a command to AMPS that returns an acknowledgement when all previous commands from this client have been processed. This command helps applications that use AMPS determine when AMPS has received all of the messages that have been sent, making it safe for the client to exit.

Header Fields

Table 26 contains the header fields available to a `flush` command.

Table 26. Header fields supported by `flush`

Field	Description
`Command`	The command to be executed. Value: `flush`.
`ClientName`	A string identifier used to give a client a unique id.
`AckType`	Acknowledgment type for the given command. Value is a comma separated list of one or more of the following: `none`, `completed` or `processed`.

Returns

A `flush` message specifying an `AckType` of `completed` or `processed` will receive an `ack` message when all previous messages from this client have been processed by AMPS.

Table 27 contains the acknowledgment messages that can be returned by a `logon` command.

Table 27. Acknowledgment messages supported by `logon`

Acknowledgment	Description
`none`	No `ack` message is returned. This is the default behavior.
`completed`	All previous commands have been processed by AMPS.
`persisted`	Not supported at this time.
`processed`	AMPS has processed the `flush` message.
`received`	The `flush` command has been received.
`stats`	Not supported at this time.

heartbeat

Description

Sends a command to AMPS that starts or refreshes a heartbeat timer. When a heartbeat timer is active, AMPS publishes periodic heartbeat messages to AMPS and expects the client to respond with a heartbeat message. If the client does not provide a heartbeat within the time specified, AMPS logs an error and disconnects the connection.

Header Fields

Table 28 contains the header fields available to a `heartbeat` command.

Table 28. Header fields supported by `hearbeat`

Field	Description
`Command`	The command to be executed. Value: `heartbeat`.
`Options`	Specifies whether this command starts the timer or refreshes the timer. Valid options are: • `start`, immediately followed by an interval. This option specifies that the command starts a timer, and sets the interval at which AMPS will expect heartbeat messages. For example, to specify an interval of 5 seconds, the option is `start,5` • `beat`. This option specifies that the command refreshes the heartbeat timer.

Returns

The `heartbeat` message does not typically request an acknowledgement, and therefore does not receive a response. The command can, however, request acknowledgements as listed below.

Table 29. Acknowledgment messages supported by `heartbeat`

Acknowledgement	Description
`none`	Not supported at this time.
`completed`	Not supported at this time.
`parsed`	Not supported at this time.
`persisted`	Not supported at this time.
`processed`	AMPS has processed the `heartbeat` message.
`received`	AMPS has received the `heartbeat` message.
`stats`	Not supported at this time.

start_timer

Description

The `start_timer` resets the AMPS client and latency statistics counter, and starts the timer for the client and latency statistic counter. This is used in conjunction with `stop_timer` to generate statistics published to the AMPS log.

Header Fields

Table 30. Header fields supported by `start_timer`

Field	Description
`Command`	Command to be executed. Value: `start_timer`.

stop_timer

Description

The `stop_timer` command is used in conjunction with the `start_timer`. Once a `stop_timer` command has been issued, AMPS will write statistics to the log file to generate a profile of the number of clients and messages observed and processed by AMPS.

Header Fields

Table 31. Header fields supported by `stop_timer`

Field	Description
`Command`	Command to be executed. Value: `stop_timer`.
`AckType`	Acknowledgment type for the given command. Only `processed` is supported for this command.
`CommandId`	If specified with an AMPS command requesting an acknowledgment message, all requested acknowledgment messages will contain the `CommandId` in the `ack` response header.

Returns

The `stop_timer` command can return an acknowledgment message if the `AckType` is set to `processed` in the `stop_timer` command message header. The returned acknowledgment message will include a `Status` field containing any messages from AMPS.

The `stop_timer` will return a `ClientStatus` message that contains the statistics measured since the last call to the `start_timer` command. Table 32 describes the `AckType` messages supported by the stop timer command.

Table 32. Acknowledgment messages supported by `stop_timer`

Acknowledgement	Description
`none`	Not supported at this time.
`completed`	Not supported at this time.
`parsed`	Not supported at this time.
`persisted`	Not supported at this time.
`processed`	AMPS has processed the `stop_timer` message.
`received`	Not supported at this time.
`stats`	Not supported at this time.

Response Messages

Content Messages

AMPS provides three types of message that contain message content

- `publish` messages return data from a topic as it is published, in order, whether the data is being published live, or is the result of a replay
- `sow` messages return data from a SOW query. These messages return the state of messages that are current as of the time for the query. By default, the messages are returned without regard to the order in which the messages were published. A query can specify the order of the returned messages based on the data within the message by including the OrderBy header on the SOW query.
- `oof` messages indicate that a content message no longer matches a subscription. These messages are sent to a client in order.

publish message

Description

AMPS returns a `publish` message to a client when a new message is published to AMPS that matches one of the subscriptions requested by the client. There are two ways that AMPS can generate publish messages:

- *Single-origin* messages. For subscriptions to topics where AMPS can identify a single source for a publish message, AMPS provides information from that publish message to the subscriber. This applies to subscriptions to unpersisted topics, SOW topics, and conflated topic replicas. This does not include subscriptions to views (or conflated topics based on views), since views provide the ability to join multiple topics and aggregate over multiple messages. For conflated topic replicas, the header information provided is the information provided with the message published to the subscriber. For messages produced by delta publish, AMPS will use the information provided on the delta publish except as noted in the table below.
- *Synthetic* messages. In some cases, AMPS must provide a message that is constructed by the server. This happens for views, and for status messages from AMPS.

AMPS provides different values in the header fields depending on the origin of the publish message. For synthetic messages, AMPS does not provide information on the origin of the message, since there may be multiple sources of the message or, in the case of status messages, no external source. Likewise, AMPS does not provide a `CorrelationId`, since that header is set by the publisher for a specific message.

Header Fields

Table 33 contains the header fields returned in a `publish` message.

Table 33. Header fields returned in a `publish` response

Field	Description
`Command`	Type of message. Always `publish`, as encoded by the protocol.
`Topic`	The topic the message was published to.
`CorrelationId`	A publisher-provided string that is passed, verbatim, to subscribers. If this header is not present, or the message is a synthetic message as described above, subscribers receive no value for the `CorrelationId`. The contents of this header must consist of characters that are legal in Base64 encoding. For delta publishes, AMPS uses the `CorrelationId` of the delta publish if one is present. If no `CorrelationId` is present on the publish, AMPS uses the `CorrelationId` of the existing message, if one is present. If there is no `CorrelationId` on the publish, and there is no `CorrelationId` for the existing message, AMPS does not provide a `CorrelationId`.
`UserId`	The `UserId` of the client that published the message. An authentication module may choose whether to allow subscribers to receive this information.
`SubIds`	The set of subscription IDs that produced this message. When a message matches multiple subscriptions, AMPS may produce a list of subscription IDs for all matching subscriptions. This header is provided by AMPS. The AMPS Clients process this list and provide a single SubscriptionID for each message provided to message handlers.
`Bookmark`	The bookmark assigned to this message, if the message was persisted to a transaction log.
`TransmissionTime`	An ISO-8601 datetime that notes the time the message was processed by AMPS. This header is included if the client requested transmission time for the subscription.
`LeasePeriod`	For messages received from a queue, the ISO-8601 datetime that indicates when the lease expires.
`SowKey`	If the message was from a topic that uses a SOW, the message includes the `SowKey` that AMPS uses to uniquely identify the message within the SOW.

sow message

Description

The `sow` message returns a record from the SOW. For more information, see the *State of the World* and *SOW Queries* chapters in the *AMPS User Guide*

Header Fields

Table 34. Header fields supported by `sow`

Field	Description
Command	Type of message. Always `sow`, as encoded by the protocol.
`Topic`	The topic from which the records were produced.
`SowKey`	An AMPS-created identifier for this message.
`BatchSize`	The number of records returned in a single `sow` batch.
`Timestamp`	The time at which AMPS generated this message.
`QueryId`	The QueryId of the query that produced this message.
`MsgLen`	The length of the first SOW message in the data portion of this message.

Data

The `sow` message contains data. The data for the message consists of up to `BatchSize` messages, formatted as expected by the protocol. Each message contains its own header, with the following fields:

Table 35. Header fields for messages in `sow` data

Field	Description
`SowKey`	An AMPS-created identifier for this message.
`CorrelationId`	A user-provided string that will be passed, verbatim, to subscribers. If this header is not present on the SOW record, subscribers receive no value for the `CorrelationId`. The contents of this header must consist of characters that are legal in Base64 encoding.
`MsgLen`	The length of the next SOW message in the data portion of this message.

oof message

Description

The `oof` message indicates that a previously-received message is no longer in focus. For more information, see the *State of the World* and *SOW Queries* chapters in the *AMPS User Guide*

Header Fields

Table 36. Header fields provided in `oof`

Field	Description
`Command`	Type of message. Always `oof`, as encoded by the protocol.
`Topic`	The topic which contained the message that has gone out of focus.
`SowKey`	An AMPS-created identifier for the message that has gone out of focus.
`Reason`	The reason the message has gone out of focus. Valid reasons include `deleted`, `expired`, `filter`, and `entitlement`.
`SubIds`	The Subscription Ids of the subscriptions that produced this message. The AMPS clients will provide this message to the handler registered for each of the subscriptions specified.
`CorrelationId`	A user-provided string that will be passed, verbatim, to subscribers. If this header is not present on the SOW record that was deleted, subscribers receive no value for the `CorrelationId`. The contents of this header must consist of characters that are legal in Base64 encoding.

Data

The `oof` message contains the updated message that caused the message to go out of focus, except if the reason is `entitlement`.

Ack Messages

AMPS provides `ack` messages to report the status of commands delivered to AMPS.

ack message

Description

The `ack` message returns status information from AMPS.

AMPS does not create `ack` messages unless an acknowledgement is specifically requested. The exact meaning and content of ack messages depends on the command the requests the message. AMPS supports the following types of ack messages with the general semantics described below.

Table 37. Types of `ack` message

ack Type	Meaning
`completed`	An operation has completed. For example, subscriptions that replay from the transaction log can produce a `completed` acknowledgement to indicate when transaction log replay has finished and further messages for the subscription are the result of new publishes.
`persisted`	Data has been persisted.

ack Type	Meaning
`processed`	AMPS has processed the command. Notice that, depending on the command, AMPS may not have executed the command when this acknowledgement is produced.
`received`	AMPS has received the command, but has not yet processed it.
`stats`	Statistics for the command. This acknowledgement is typically produced after the command has fully completed.

Common Header Fields For Ack Messages

Table 38. Header fields provided in `ack`

Field	Description
`Command`	Type of message. Always `ack`, as encoded by the protocol.
`AckType`	The type of acknowledgment. One of `completed`, `persisted`, `processed`, `received` or `stats`.
`CommandId`	The `CommandId` that this ack refers to. Clients can use this field to correlate the `ack` returned with the command being acknowledged.
`Status`	The status of the command.
`Reason`	The reason for a `failure` status.

Additional fields for logon

When the `ack` message is produced in response to a `logon` command, the following additional header fields may be set:

Table 39. Additional `ack` headers for `logon`

Field	Description
`ClientName`	The name of the client provided with the command.
`SequenceId`	The last `SequenceId` persisted to the transaction log for this client, as identified by the `ClientName`.
`Bookmark`	The last bookmark from this client.
`UserId`	`UserId` to use when the status is retry.
`Password`	`Password` to use when the status is retry.
`Version`	The version of the AMPS server.

Additional fields for publish and delta_publish

When the `ack` message is produced in response to a `publish` or `delta_publish` command, the following additional header fields may be set:

Table 40. Additional `ack` headers for `publish` or `delta_publish`

Field	Description
`SequenceId`	The last SequenceId persisted for this client.

Field	Description
`Bookmark`	The last Bookmark persisted for this client.

Additional fields for subscribe and delta_subscribe

When the `ack` message is produced in response to a `subscribe` or `delta_subscribe` command, the following additional header fields may be set:

Table 41. Additional `ack` headers for `subscribe` or `delta_subscribe`

Field	Description
`SubId`	The `SubId` sent with the command, or the `SubId` generated by AMPS if no `SubId` was provided. This field is not returned in `processed` acks.
`Options`	Returned when the command is a `subscribe` to a queue. Contains the following options: `max_backlog` indicates the effective maximum backlog that the server has assigned for this subscription.

Additional fields for unsubscribe

When the `ack` message is produced in response to a `unsubscribe`, AMPS does not provide additional header fields.

Additional fields for sow, sow_and_subscribe, sow_and_delta_subscribe

When the `ack` message is produced in response to a `sow`, `sow_and_subscribe`, or `sow_and_delta_subscribe`, the following additional header fields may be set:

Table 42. Additional `ack` headers for `sow, sow_and_subscribe, sow_and_delta_subscribe`

Field	Description
`SubId`	The `SubId` sent with the sow command.
`QueryId`	The `QueryId` sent with the sow command.
`RecordsReturned`	The number of records returned by a SOW query. This header field is present on `stats` acknowledgements.
`TopicMatches`	The total number of records compared across all matching SOW topics. This header field is present on `stats` acknowledgements.
`Matches`	The number of records returned that match the topic regular expression and content filter. This header field is present on `stats` acknowledgements.

Additional fields for sow_delete

When the `ack` message is produced in response to a `sow_delete` the following additional header fields may be set:

Table 43. Additional `ack` headers for `sow_delete`

Field	Description
`QueryId`	The `QueryId` sent with the `sow_delete` command.
`RecordsDeleted`	The number of records deleted by the command. This header field is present on `stats` acknowledgements.
`TopicMatches`	The total number of records compared across all matching SOW topics. This header field is present on `stats` acknowledgements.
`Matches`	The number of records returned that match the topic regular expression and content filter. This header field is present on `stats` acknowledgements.

Additional fields for stop_timer

When the `ack` message is produced in response to a `stop_timer` the following additional header fields may be set:

Table 44. Additional `ack` headers for `stop_timer`

Field	Description
`Data`	The content of the message body. The message body can contain the following data: • `elapsed_time` • `mean` • `median` • `max` • `mean` • `nintieth` • `ninety_fifth` • `ninety_ninth` • `std_deviation` • `byte_count` • `match_count` • `publish_count`
`TransmissionTime`	An ISO-8601 date-time code indicating when the message is sent by the client. Used only if set on incoming message.
`TopicMatches`	The total number of records compared across all matching SOW topics. This header field is present on `stats` acknowledgements.
`Matches`	The number of records returned that match the topic regular expression and content filter. This header field is present on `stats` acknowledgements.

Data

The `ack` message does not contain data.

Query Delimiters

AMPS provides a pair of delimiters, `group_begin` and `group_end`, that indicate when a query batch begins and ends.

group_begin message

Description

The `group_begin` message marks the beginning of a set of records returned by a SOW query. For more information, see the *State of the World* and *SOW Queries* chapters in the *AMPS User Guide*

Header Fields

Table 45. Header fields provided in `group_begin`

Field	Description
`Command`	Type of message. Always `group_begin`, as encoded by the protocol.
`QueryId`	The QueryId of the query that produced this message.

group_end message

Description

The `group_end` message marks the end of a set of records returned by a SOW query. For more information, see the *State of the World* and *SOW Queries* chapters in the *AMPS User Guide*

Header Fields

Table 46. Header fields provided in `group_end`

Field	Description
`Command`	Type of message. Always `group_end`, as encoded by the protocol.
`QueryId`	The QueryId of the query that produced this message.

3. Protocol Reference

This section contains information on how different protocols represent AMPS headers. The AMPS clients handle constructing and parsing AMPS headers. However, understanding the format of command can be useful when inspecting trace level logs or network traffic captures.

FIX/NVFIX protocol

FIX/NVFIX Message Header - Sorted by Value

FIX/NVFIX Header Field	Name
20000	`Command`
20001	`CommandId`
20002	`ClientName`
20003	`UserId`
20004	`TransmissionTime`
20005	`Topic`
20006	`Filter`
20007	`MessageType`
20008	`AckType`
20009	`SubscriptionId`
20011	`Version`
20012	`Expiration`
20013	`SendSubscriptionIDs`
20014	`DataOnly`
20015	`Heartbeat`
20016	`TimeoutInterval`
20017	`LeasePeriod`
20018	`Status`
20019	`QueryID`
20020	`SendOutOfFocus`
20021	`LogLevel`
20022	`UseNamespaces`
20023	`BatchSize`
20025	`TopNRecordsReturned`
20029	`SendEmpty`
20031	`MaximumMessages`
20032	`SowKeys`
20033	`SendKeys`
20034	`Src`
20035	`CorrelationId`
20036	`Sequence`
20037	`Bookmark`

FIX/NVFIX Header Field	Name
20038	`Password`
20039	`Options`
20052	`RecordsInserted`
20053	`RecordsUpdated`
20054	`RecordsDeleted`
20055	`RecordsReturned`
20056	`TopicMatches`
20057	`Matches`
20058	`MessageLength`
20059	`SowKey`
20060	`GroupSequenceNumber`
20061	`SubscriptionIds`
20062	`Reason`
20063	`MessageID`
20074	`CorrelationID`

FIX/NVFIX Message Header - Sorted by Name

FIX/NVFIX Header Field	Name
20008	`AckTyp`
20037	`BkMrk`
20023	`BtchSz`
20002	`ClntName`
20000	`Cmd`
20001	`CmdId`
20035	`CorrelationId`
20014	`DatOnly`
20012	`Expn`
20006	`Fltr`
20017	`GrcPrd`
20060	`GrpSqNum`
20015	`Hrtbt`
20017	`LeasePeriod`
20021	`LogLvl`
20057	`Matches`
20063	`MsgId`

FIX/NVFIX Header Field	Name
20058	`MsgLen`
20007	`MsgTyp`
20031	`MxMsgs`
20039	`Opts`
20038	`PW`
20019	`Qld`
20062	`Reason`
20054	`RecordsDeleted`
20053	`RecordsInserted`
20055	`RecordsReturned`
20036	`Seq`
20029	`SndEmpty`
20033	`SndKeys`
20020	`SndOOF`
20013	`SndSubIds`
20059	`SowKey`
20032	`SowKeys`
20034	`Src`
20018	`Status`
20009	`SubId`
20061	`SubIds`
20016	`TmIntvl`
20025	`TopN`
20056	`TopicMatches`
20005	`Tpc`
20004	`TxmTm`
20022	`UseNs`
20003	`UsrId`

XML Protocol

XML Message Header - Sorted by Name

XML Header Field	Name
AckTyp	`AckType`

XML Header Field	Name
BkMrk	`Bookmark`
BtchSz	`BatchSize`
ClntName	`ClientName`
Cmd	`Command`
CmdId	`CommandId`
DatOnly	`DataOnly`
Expn	`Expiration`
Fltr	`Filter`
GrcPrd	`GracePeriod`
GrpSqNum	`GroupSequenceNumber`
Hrtbt	`Heartbeat`
LeasePeriod	`LeasePeriod`
LogLvl	`LogLevel`
Matches	`Matches`
MsgId	`MessageID`
MsgLen	`MessageLength`
MsgTyp	`MessageType`
MxMsgs	`MaximumMessages`
Opts	`Opts`
PW	`Password`
QId	`QueryID`
Reason	`Reason`
RecordsDeleted	`RecordsDeleted`
RecordsReturned	`RecordsReturned`
Seq	`Sequence`
SndEmpty	`SendEmpty`
SndKeys	`SendKeys`
SndOOF	`SendOutOfFocus`
SndSubIds	`SendSubscriptionIDs`
SowKey	`SowKey`
SowKeys	`SowKeys`
Status	`Status`
SubId	`SubscriptionId`
SubIds	`SubscriptionIds`
TmIntvl	`TimeoutInterval`
TopN	`TopNRecordsReturned`

XML Header Field	Name
TopicMatches	`TopicMatches`
Tpc	`Topic`
TxmTm	`TransmissionTime`
UseNS	`UseNamespaces`
UsrId	`UserId`

AMPS/JSON Protocol

AMPS Message Header - Sorted by Name

AMPS Header Field	Abbreviation	Name
ack_type	a	`AckType`
password	pw	`Password`
bookmark	bm	`Bookmark`
batch_size	bs	`BatchSize`
client_name		`ClientName`
cmd	c	`Command`
cmd_id	cid	`CommandId`
correlation_id	x	`CorrelationId`
data_only		`DataOnly`
expiration	e	`Expiration`
filter	f	`Filter`
gseq		`GroupSequenceNumber`
heartbeat		`Heartbeat`
leaseperiod	lp	`LeasePeriod`
matches		`Matches`
msg_len	l	`MessageLength`
max_msgs		`MaximumMessages`
opts	o	`Opts`
orderby		`OrderBy`
query_id		`QueryID`
reason		`Reason`
records_deleted		`RecordsDeleted`
records_inserted		`RecordsInserted`
records_returned		`RecordsReturned`

AMPS Header Field	Abbreviation	Name
records_updated		`RecordsUpdated`
seq	s	`Sequence`
send_empty		`SendEmpty`
send_keys		`SendKeys`
send_oof		`SendOutOfFocus`
sow_key	k	`SowKey`
sow_keys		`SowKeys`
status		`Status`
sub_id		`SubscriptionId`
sids		`SubscriptionIds`
src		`Src`
timeout_interval		`TimeoutInterval`
timestamp	ts	`Timestamp`
top_n		`TopNRecordsReturned`
topic_matches		`TopicMatches`
topic	t	`Topic`
use_ns		`UseNamespaces`
user_id		`UserId`
version	v	`Version`

Header Fields - Reference

Name	Type	Definition
`AckType`	string	Acknowledgement type for the given command.
`BatchSize`	Integer. Default is 1 when not present.	Specifies the number of messages that are batched together when returning a query result.
`Bookmark`	string	A client-originated identifier used to mark a location in journaled messages.
`ClientName`	string	Used to identify a client. Useful for publishers that wish to identify the source of a publish, client status messages and for client heartbeats. Can be set with `logon` command.
`Command`	One of: • `publish`	Command to be executed.

Name	Type	Definition
	• `subscribe` • `sow` • `sow_and_subscribe` • `sow_delete` • `unsubscribe` • `flush` • `heartbeat` • `start_timer` • `stop_timer` • `logon`	
`CommandId`	string	Client-specified command id. The `CmdId` is returned by the engine in responses to commands to allow the client to correlate the response to the command.
`CorrelationId`	string, base64 encoded characters only	Opaque token set by an application and returned with the message.
`DataOnly`	Boolean (`true` or `false`)	If `true`, only send raw data to subscriber for a matching publish message, i.e. do not include FIX/NVFIX envelope.
`Expiration`	integer (seconds)	SOW expiration time if used in `publish`.
`Filter`	string, should wrap in CDATA	Content filter expression.
`GracePeriod`	integer (milliseconds)	Grace period after heartbeat interval is exceeded before client is considered in error state.
`GroupSequenceNumber`	integer	Group Sequence Number returned with each batch message of a SOW response.
`Heartbeat`	one of: `start`, `stop`, `beat`	Heartbeat command.
`LeasePeriod`	timestamp	For messages from a queue, the time at which the lease expires.
`LogLevel`	one of: `info`, `none`	Set the log level.
`Matches`	integer	Returned in the acknowledgement to a SOW query that indicates number of matches.
`MaximumMessages`	integer greater than zero	Specifies the maximum number of messages within a batch publish.

Name	Type	Definition
`MessageID`	string, e.g. `MAMPS-XYZ`	Set by AMPS engine to tag every incoming message.
`MessageLength`	integer	Sent with XML formatted message data to indicate the number of bytes used by the message body.
`MessageType`	one of: `xml`, `fix`, `nvfix`	Message type.
`Opts`	string	A comma-delimited list of options on a specific command.
`Password`	string	Password used to authenticate with an AMPS server.
`QueryID`	string	SOW Query identifier set by client to identify a query.
`Reason`	string	The failure message that appears when an acknowledgement returns a `status` of `failure`.
`RecordsDeleted`	integer	Used in conjunction with the `stats` acknowledgement, this is the number of records deleted from the SOW with a `sow_delete` command.
`RecordsInserted`	integer	Used in conjunction with the `stats` acknowledgement, this is the number of records inserted into the SOW.
`RecordsUpdated`	integer	Used in conjunction with the `stats` acknowledgement, this is the number of records updated in the SOW.
`RecordsReturned`	integer	Returned in the acknowledgement to an SOW query that indicates number of records in the store.
`SendEmpty`	Boolean (`true` or `false`); default is `true`	If `true`, empty messages that are published will be forwarded to matching subscriptions.
`SendKeys`	Boolean (`true` or `false`)	Option to instruct AMPS that a client would like to receive the SowKey(s) back.
`SendOutOfFocus`	Boolean (`true` or `false`)	If `true`, Out-of-Focus messages are sent for the SOW query.
`SendSubscriptionIDs`	Boolean (`true` or `false`)	If `false`, subscription identifiers will not be sent for a matched message.
`Sequence`	integer greater than zero	An integer that corresponds to the publish message sequence number. For more information see the Replication section in the User Guide.
`SowKey`	unsigned long	An `SowKey` will accompany each message returned in an SOW batch.

Name	Type	Definition
		For XML it will be contained in the `Msg` section. A `SowKey` may also be added to messages coming in on a subscription when the published message matches a record in the SOW.
`SowKeys`	integers	Comma-separated list of `SowKey` integers.
`Status`	one of: `stopped`, `alive`, `timed out`, `error`	Used to indicate client status when client is monitored for heartbeats.
`SubscriptionId`	string, e.g. `SAMPS-XYZ`	The subscription identifier set by server when processing a subscription.
`SubscriptionIds`	string	Comma-separated list of `SubIds` sent from AMPS engine to identify which client subscriptions match a given publish message.
`TimeoutInterval`	integer	Used in conjunction with the heartbeat interval to set the timeout interval for a publisher.
`TopNRecordsReturned`	unsigned integer	The number of records to return. Note: If `TopN` is not equally divisible by the `BtchSz`, then more records will be returned so that the total number of records is equally divisible by the `BtchSz` setting.
`Topic`	string	Topic
`TopicMatches`	integer	Returned in the acknowledgement to an SOW query that indicates number of topic matches.
`TransmissionTime`	ISO-8601 date-time	Transmission timestamp set by client.
`UseNamespaces`	Boolean (`true` or `false`)	Use SOAP XML namespaces in all messages from the AMPS engine.
`UserId`	string	Used to identify the user id of a command.
`Version`	string	Contains the version of the AMPS server.