AMPS Configuration Reference Guide

AMPS Configuration Reference Guide

5.2

Publication date Jun 26, 2017

Table of Contents

Chapter 1. AMPS Configuration Basics

If you have not become familiar with the *AMPS User Guide*, in particular the *Getting Started* chapter, please start there before reading this guide.

The easiest way to create a custom XML configuration file for AMPS is to start with the sample configuration file produced by the `--sample-config` flag to AMPS. Example 1.1 shows a simplified sample configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Sample AMPS configuration

     This file defines an AMPS instance that provides publish and
     subscribe, topic filtering, and content filtering for JSON messages.
     The instance provides messaging services on port 9007 of the server.
     This configuration also provides an adminstrative interface on
     port 8085, and logs serious messages (error and higher severity) to
     stdout.

     This sample file does not configure State of the World (SOW) Topics,
     Transaction Logs, Aggregation and Views, Historical Query, Replication,
     Authentication and Entitlement, Conflating Topic Replicas, or other
     features of AMPS.

     More details for the features available and how to configure them are
     provided in the AMPS User Guide and the AMPS Configuration Reference.
     Both are available at http://crankuptheamps.com/documentation/

  -->

<AMPSConfig>

  <!-- Name of the AMPS instance -->

  <Name>AMPS-Sample</Name>

  <!-- Configure the administrative HTTP server on port 8085

       This HTTP server provides admin functions and statistics
       for the instance
   -->

  <Admin>
    <InetAddr>localhost:8085</InetAddr>
  </Admin>

  <!-- Configure a transport for JSON messages over TCP on port 9007
    -->

  <Transports>
    <Transport>
      <Name>json-tcp</Name>
      <Type>tcp</Type>
      <InetAddr>9007</InetAddr>
      <MessageType>json</MessageType>
      <Protocol>amps</Protocol>
    </Transport>
  </Transports>

 <!-- Log messages of severity 'error' and higher to stdout -->

  <Logging>
    <Target>
      <Protocol>stdout</Protocol>
      <Level>error</Level>
    </Target>
  </Logging>

</AMPSConfig>
```

Example 1.1. Simple AMPS configuration file

The AMPS configuration XML file is defined first by wrapping the config file with an `AMPSConfig` tag to identify it as a configuration file. Next, the instance is given a name using the `<Name>` tag.

Once our instance has a name, it is good to define the connection target for the administration port. By default, the administration port can be found by pointing a browser to `http://localhost:8085`, but if a different port or host name is desired, then that is defined in the `Admin` and `InetAddr` tags. The Admin port is discussed more in Table 4.1.

Next we describe how to get messages into AMPS. There are several different transport types which can be parsed by AMPS, all of which are discussed in greater detail in the Transports chapter, but for this sample, we keep things simple by focusing on JSON messages over `tcp`. In AMPS, each key to defining each transport is to give them a unique `InetAddr` port and specify the type of message AMPS will process on that port using `MessageType` tag. The `MessageType` tells AMPS how to parse the incoming messages on a specific port. In the above example, messages are coming in on port 9007, and AMPS uses the JSON parser to parse the body of the message. AMPS also requires a `Protocol` tag for the `Transport`, which specifies the format of the commands to AMPS. In this case, we use the standard `amps` protocol. (Older AMPS applications and AMPS installations may require a different protocol format, such as `fix` or `xml`. There's no functional difference between these protocols, but the AMPS server and the clients need to use the same protocol format to successfully exchange messages.)

The last portion of the configuration is Logging. In the above example, the `Logging` tag defines only one log target, but it's quite common to have one or more `Logging` targets. Again referring to the example, all logging messages that are at `error` level and above will be logged to the logging `Protocol` of `stdout`. In other words, these messages will be logged to the terminal, and not to a file. AMPS supports a robust set of logging features and configurations, all of which are covered in more detail in the *Logging* chapter in the *AMPS User Guide* and in Chapter 8 of this reference.

1.1. AMPS Configuration File Special Characters

In AMPS there are a few special characters that you should be aware of when creating your configuration file. These characters can provide some handy short cuts and make configuration creation easier, but you should also be aware of them so as not to introduce errors.

State of the World File Name

When specifying the file for a State of the World database, using the `%n` string in the file name specifies that the AMPS server will use the message type and topic name in that position to create a unique filename. Example 1.2 shows how to use this in the AMPS configuration file.

```
 <SOW>
    <Topic>
      <Topic>Customers</Topic>
      <FileName>./sow/%n.sow</FileName>
      <MessageType>json</MessageType>
      <Key>/customerId</Key>
    </Topic>
</SOW>
```

Example 1.2. SOW file name tokens used in configuration file

Log Rotation Name

When specifying an AMPS log file which has `RotationThreshold` specified, using the `%n` string in the log file name is a useful mechanism for ensuring the name of the log file is unique and sequential. Example 1.3 shows a file name token replacement in the AMPS configuration file.

```
<Logging>
  <Target>
    <Protocol>file</Protocol>
    <Level>info</Level>
    <FileName>log/log-%n.log</FileName>
    <RotationThreshold>2G</RotationThreshold>
  </Target>
</Logging>
```

Example 1.3. Log file name tokens used in configuration file

In the above example, a log file will be created in the `AMPSDIR/log/` directory. The first time this file is created, it will be named `log-1.log`. Once the log file reaches the `RotationThreshold` limit of 2G, the previous log file will be saved, and the new log file name will be incremented by one. Thus, the next log file will be named `AMPSDIR/log/log-2.log`.

Dates

AMPS allows administrators to use date-based file names when specifying the file name in the configuration, as demonstrated in Example 1.4.

```
<Logging>
  <Target>
```

```
    <Protocol>file</Protocol>
    <Level>info</Level>
    <FileName>
      log/log-%Y-%m-%dT%H%M%S.log
    </FileName>
    <RotationThreshold>2G</RotationThreshold>
  </Target>
</Logging>
```

Example 1.4. Date tokens used in configuration file

In the above example, a log file will be created in the `$AMPSDIR/log` named `2011-01-01-120000.log` if the log was created at noon on January 1, 2011.

AMPS provides full support for the date tokens provided by the standard strftime function, with the exception of `%n`, as described above. The following table shows some of the most commonly used tokens:

Table 1.1. Commonly Used Date and Time Tokens

Token	Provides	Example
%a	Short weekday name	Fri
%A	Full weekday name	Friday
%b	Short month name	Feb
%B	Full month name	February
%c	Simple date and time	Fri Feb 14 17:25:00 2014
%C	Century	20
%d	Day of the month (leading zero if necessary)	05
%D	Short date format (MM/DD/YY)	02/20/14
%e	Day of the month (leading space if necessary)	5
%F	Short date format (YYYY-MM-DD)	2014-02-20
%H	Hour (00-23)	17
%I	Hour (00-12)	05
%j	Day of the year (001-366)	051
%m	Month (01-12)	02
%p	AM or PM	PM
%r	Current time, 12 hour format	05:25:00 pm
%R	Current time, 24 hour format	17:25
%T	ISO 8601 Time format	17:25:00
%u	ISO 8601 day of the week (1-7, Monday = 1)	5
%V	ISO 8601 week number (00-53)	07
%y	Year, last two digits	14
%Y	Year, four digits	2014
%Z	Timezone name or abbreviation (blank if undetermined)	PST

1.2. Using Units in the Configuration

To make configuration easy, AMPS permits the use of units to expand values. For example, if a time interval is measured in seconds, then the letter `s` can be appended to the value. For example, the following SOW topic definition used the `Expiration` tag to set the record expiration to 86400 seconds (one day).

```
<SOW>
  <Topic>
    ...
    <Expiration>86400s </Expiration>
    ...
  </Topic>
</SOW>
```

Example 1.5. Expiration Using Seconds

An even easier way to specify an expiration of one day is to use the following `Expiration`:

```
<SOW>
  <Topic>
     ...
     <Expiration>1d</Expiration>
     ...
  </Topic>
</SOW>
```

Example 1.6. Expiration Using Days

Table 1.2 shows a listing of the time units AMPS supports in the configuration file.

Table 1.2. AMPS Configuration - Time Units

Units	Description
`ns`	nanoseconds
`us`	microseconds
`ms`	milliseconds
`s`	seconds
`m`	minutes
`h`	hours
`d`	days
`w`	weeks

AMPS configuration supports a similar mechanism for byte-based units when specifying sizes in the configuration file. Table 1.3 shows a listing of the byte units AMPS supports in the configuration file.

Table 1.3. AMPS Configuration - Byte Units

Units	Description
`kb`	kilobytes
`mb`	megabytes

Units	Description
`gb`	gigabytes
`tb`	terabytes

Dealing with large numbers in AMPS configuration can also be simplified by using common exponent values to handle raw values. This means that instead of having to input `10000000` to represent ten million, a user can input `10M`. Table 1.4 contains a list of the exponents supported.

Table 1.4. AMPS Configuration - Numeric Units

Units	Description
k	10^3 - thousand
M	10^6 - million

To make it easier for users to remember the units, AMPS interval and byte units are not case sensitive.

1.3. Environment Variables in AMPS Configuration

AMPS configuration also allows for environment variables to be used as part of the data when specifying a configuration file. These variables can be set in the environment when AMPS starts, or passed to AMPS using the `-D` option on the command line.

If a global system variable is commonly used in an organization, then it may be useful to define this in one location and re-use it across multiple AMPS installations or applications. AMPS will replace any token wrapped in `${}` with the environment variable defined in the current user operating system environment. Example 1.7 demonstrates how the environment variable `ENV_LOG` is used to define a global environment variable for the location of the host logging.

```
<Logging>
  <Target>
    <Protocol>file</Protocol>
    <FileName>${ENV_LOG}</FileName>
    <Level>info</Level>
    <RotationThreshold>2G</RotationThreshold>
  </Target>
</Logging>
```

Example 1.7. Environment Variable Used in Configuration

Internal Environment Variables

In addition to supporting custom environment variables, AMPS includes a configuration variable, `AMPS_CONFIG_DIRECTORY`, which can be used to reference the directory in which the configuration file used to start AMPS is located. For example, assume that AMPS was started with the following command at the command prompt:

```
%>./ampServer ../amps/config/config.xml
```

Given this command, the log file configuration option shown in Example 1.8 can be used to instruct AMPS to create the log files in the same parent directory as the configuration file — in this case `../amps/config/logs/infoLog.log`.

```
<Logging>
  <Target>
    <Protocol>file</Protocol>
    <FileName>
      ${AMPS_CONFIG_DIRECTORY}/logs/infoLog.log
    </FileName>
    <Level>info</Level>
    <RotationThreshold>2G</RotationThreshold>
  </Target>
</Logging>
```

Example 1.8. AMPS_CONFIG_DIRECTORY Environment Variable Example

In addition to the `AMPS_CONFIG_DIRECTORY` environment variable, AMPS also supports the `AMPS_CONFIG_PATH`, which is an absolute path to the configuration file used to start AMPS.

1.4. Including External Files

For production applications, AMPS configuration files can become large and complicated. In many cases, different instances of an AMPS server need to reuse the same definitions. For example, both servers in a High-Availability pair may need to use the same queue and SOW definitions.

To help you manage complicated configurations and more easily keep configuration consistent on different servers, AMPS allows you to include external files in the configuration file by using the `Include` directive.

For example, you could use this for a High-Availability pair to include a file that defines the queue, transaction log, and topic definitions. Both instances could include exactly the same file for those definitions, while having different instance names and port numbers.

When AMPS loads a configuration file that contains an Include directive, AMPS follows this process:

- Load and parse the configuration file
- If the file contains any `Include` directives, load and parse the files specified by those directives. If the included files contain Include directives, load and parse the files specified by those directives (and so forth until all Include directives have been processed).
- Once all files have been loaded and parsed, replace the Include directives in the original files with the parsed files.

AMPS does not process the configuration file until all of the `Include` directives have been resolved and the files have been parsed.

A file may not be included by any file that it includes, or it is impossible for AMPS to complete the parsing process.

Because each file is individually parsed, XML entities defined in a file are not defined for the files that are `Include`d by that file.

To make it easier to identify which elements of the complete AMPS configuration file have been inserted through the `Include` mechanism, AMPS can include comments in the assembled file that indicate the source file for configuration elements. By default, this feature is off, and XML content is included verbatim. To change the default for the instance, use the `ConfigIncludeCommentDefault` configuration element to enable comments, by default, for every `Include` in the instance. To override commenting behavior for an individual `Include`, use the `comment` attribute.

Example

Consider a configuration file with the following `Logging` element defined:

```
<AMPSConfig>
 ...
  <Logging>
     <Include comment="true">filetarget.xml</Include>
  </Logging>
</AMPSConfig>
```

After parsing the configuration file, AMPS loads and parses the `filetarget.xml` file and replaces the Include element with the contents of that file.

Suppose `filetarget.xml` contains the following `Target` directive:

```
<Target>
   <Protocol>file</Protocol>
   <FileName>/var/log/amps-log-%n.log</FileName>
   <Level>info</Level>
</Target>
```

The configuration that AMPS uses will be effectively the same as if the configuration file contained the following XML:

```
<AMPSConfig>
  ...
  <Logging>
     <!-- Start <Include>filetarget.xml</Include> -->
     <Target>
         <Protocol>file</Protocol>
         <FileName>/var/log/amps-log-%n.log</FileName>
         <Level>info</Level>
     </Target>
     <!-- End <Include>filetarget.xml</Include> -->
</Logging>
  ...
</AMPSConfig>
```

Chapter 2. Generating a Configuration File

This appendix includes a listing of all AMPS configuration parameters. AMPS provides a command line option to help an administrator quickly set up an AMPS server. In addition to the quick setup discussed in the *Getting Started* chapter of the *AMPS User Guide*, AMPS also provides the following command line options to create a basic XML configuration file. Running the following command will create a configuration file named `config.xml`. The generated file is a bare-bones configuration that allows AMPS to start, process JSON messages, and provide monitoring through the admin interface.

```
ampServer --sample-config > config.xml
```

The AMPS server also provides the ability to perform basic validation of the config file, using the `--verify-config` flag.

```
ampServer --verify-config config.xml
```

The validation process checks for errors in the configuration that would prevent AMPS from starting, and reports warnings and informational messages about the configuration file. However, the validation process does not ensure that the configuration file provided is suitable for any particular purpose.

Chapter 3. Instance Level Configuration

This chapter describes elements of the AMPS configuration that set parameters for the instance as a whole.

Table 3.1. Instance Level Configuration Parameters

Element	Description
`Name`	This element defines the name of your AMPS instance. The instance name is used to uniquely identify this instance for replication purposes, to generate file names for use by the AMPS instance, and is shown in log statements and for other administrative purposes. 60East recommends that the name be short and meaningful, and that each instance in your AMPS installation have a distinct name. When creating a name, the name should not contain special characters such as spaces, path separator characters (`/` or `\`), or characters that will be interpreted by the Linux shell (`$` or `~`). This element is required, and there is no default.
`Group`	Identifies the replication group for this instance. If no `Group` element is present, the replication group for this instance is set to the `Name` of the instance. Set the group parameter when being able to refer to a set of instances makes your replication configuration simpler. Replication passthrough uses the group name to specify which instances to provide passthrough for. See the *AMPS User Guide* for a discussion of replication, including passthrough. Defaults to the instance `Name`.
`ProcessName`	Specifies the process name to set for this instance. When this element is present, AMPS uses the name specified as the process name. Otherwise, the process name uses the default set by Linux, which is the executable name (`ampServer` or `ampServer-compat` unless the executable has been renamed.) This element is most useful for systems that host multiple AMPS instances and want to be able to quickly tell the instances apart based on the process name. This element is optional. If not present, the AMPS executable does not change the process name.
`Regex-TopicSupport`	Sets whether this instance supports regular expression topic matching. When this option is `true`, clients can register subscriptions using regular expressions and receive messages for all matching topics. When this option is `false`, regular expression characters are interpreted as literal characters. Defaults to `true`.
`Authentica-tion`	Sets the default authentication module to use for transports that do not explicitly specify an authentication module. Authentication modules verify the identity of a connected user. The module specified must be one of the modules configured in the `Modules` element or one of the authentication modules that AMPS loads by default. See Table 12.2 for the list of default modules. Defaults to `amps-default-authentication-module`.
`Entitlement`	Sets the default entitlement module to use for transports that do not explicitly specify an entitlement module. Entitlement modules enforce permissions for a connected user. The module specified must be one of the modules configured in the `Modules` element or one of the modules that AMPS loads by default. See for Table 13.2 for the list of default modules.

Element	Description
	Defaults to: `amps-default-entitlement-module`
`Authenticator`	Sets the default authenticator module to use for outgoing connections from AMPS that do not explicitly specify an authenticator module. Authenticator modules provide credentials to use for outgoing connections. The module specified must be one of the modules configured in the `Modules` element or one of hte modules that AMPS loads by default. Defaults to: `amps-default-authenticator-module`
`SuggestedMinimumVersion`	The suggested minimum AMPS version to use this configuration file. If the AMPS instance that loads this configuration file has a version number less than the suggested minimum version, AMPS issues a warning. This option can be useful when upgrading a set of AMPS instances, or when the AMPS instance will see improved performance from a particular feature. For example, an application that will run correctly without hash indexes, but would see improved performance with hash indexes, could provide a `SuggestedMinimumVersion` of 4.3.1.0. Defaults: when no value is provided, AMPS does not check the configuration file against the verison number of the instance.
`RequiredMinimumVersion`	The required minimum AMPS version to use this configuration file. If the AMPS instance that loads this configuration file has a version number less than the suggested minimum version, AMPS issues an error and will not start. This option can be useful for enforcing upgrade on a set of AMPS instances, or when the AMPS instance must support a particular feature. For example, an application that uses message queues could provide a `RequiredMinimumVersion` of 5.0. Defaults: when no value is provided, AMPS does not check the configuration file against the verison number of the instance.
`ConfigIncludeCommentDefault`	Sets the default for how `Include` directives indicate the source of the content. When this option is set to `true` or `enabled`, content inserted through an Include directive is surrounded by comments indicating the source of the content. Defaults: Defaults to `false`, which specifies that AMPS will not surround included content with comments.
`ConfigCycleDetectionThreshold`	Sets the maximum size to allow for an expanded configuration file. This setting is intended to prevent cycles of include files (for example, where file A includes file B and file B includes file A) from consuming all of the memory on the system before failing. Defaults: Defaults to `5MB`.

```
<AMPSConfig>
    ....
    <Name>AMPS</Name>
    <Group>Sample-AMPS</Group>
    ....
</AMPSConfig>
```

Example 3.1. Instance-Level Configuration Example

3.1. SOW Statistics Interval

AMPS can publish SOW statistics for each SOW topic which has been configured. The `SOWStatsInterval` is specified as an interval (see Table 1.2) between updates to the `/AMPS/SOWStats` topic.

Table 3.2. SOW Statistics Interval Parameters

Element	Description
`SOWStatsInterval`	Interval for which SOW statistics are updated.

```
<AMPSConfig>
  ...
  <SOWStatsInterval>10s</SOWStatsInterval>
  ...
</AMPSConfig>
```

Example 3.2. SOW Statistics Interval Example

3.2. Slow Client Policies

AMPS includes a set of parameters that specify how the instance should manage slow clients. Sometimes, AMPS can publish messages faster than an individual client can consume messages, particularly in applications where the pattern of messages includes "bursts" of messages. Clients that are unable to consume messages faster or equal to the rate messages are being sent to them are ”slow clients”. By default, AMPS queues messages for a slow client in memory to grant the slow client the opportunity to catch up. However, scenarios may arise where a client can be over-subscribed to the point that the client cannot consume messages as fast as messages are being sent to it. In particular, this can happen with the results of a large SOW query, where AMPS generates all of the messages for the query much faster than the network can transmit the messages.

Slow client management is one of the ways that AMPS prevents slow clients from disrupting service to the instance. 60East recommends enabling slow client management for instances that serve high message volume or are mission critical. Slow client policies for all Transports in the instance are set at the root level of the configuration file. A Transport can override any of these settings, or choose to use the instance-wide settings. Details on slow client handling are available in the *AMPS User Guide*.

Table 3.3. Slow Client Management

Element	Description
`MessageMemoryLimit`	The total amount of memory to allocate to messages before offlining clients. This applies to all clients. For example, setting a value of `500MB` means that all clients that this limit applies to will share `500MB` for all buffered messages to those clients. Default: 10% of total host memory or 10% of the amount of host memory AMPS is allowed to consume (as reported by `ulimit -m`), whichever is *lowest*.
`MessageDiskLimit`	The total amount of disk space to allocate to messages before disconnecting clients. Default: `1GB` or the amount specified in the MessageMemoryLimit, whichever is *highest*.

Element	Description
`MessageDiskPath`	The path to use to write offline files. Default: `/var/tmp`
`ClientMessageAgeLimit`	The maximum amount of time for the client to lag behind. If a message for the client has been held longer than this time, the client will be disconnected. This parameter is an AMPS time interval (for example, `30s` for 30 seconds, or `1h` for 1 hour). Default: No age limit
`ClientMaxCapacity`	The amount of available capacity a single client can consume. Before a client is offlined, this limit applies to the `MessageMemoryLimit`. After a client is offlined, this limit applies to the `MessageDiskLimit`. This parameter is a percentage of the total. Default: `100%` (no effective limit)

3.3. Minidump Settings

AMPS minidumps contain information on the current state of the AMPS program execution, which is useful for support and diagnostics. AMPS will generate a minidump file on any crash event, or a minidump file can be generated at any point in time through the monitoring interface (see the *AMPS Monitoring Reference Guide*).

AMPS allows you to set the directory in which minidump files will be created and the permissions mask for minidump files.

Table 3.4. Mini Dump Directory Parameters

Element	**Description**
`MiniDumpDirectory`	Location to store AMPS mini dumps. Default is `/tmp`. If the directory does not exist, AMPS creates the directory. The special value `disabled` configures AMPS not to produce mini dumps.
`MiniDumpFileMask`	Permissions mask for minidump files. The value of the mask is an octal number (by convention, four digits) in the same format as the standard *umask* command, and AMPS applies this mask exactly as the *umask* command would. The file is created with the user and group that the AMPS server process runs under. AMPS accepts a maximum value of `666` for the mask. For example, here are some common umask settings:

Table 3.5. umask examples

Value	Result
`0444`	File is readable by owner, group, and any user.
`0440`	File is readable by owner and members of the owner's group

Element	Description
	<table><tr><td>Value</td><td>Result</td></tr><tr><td>`0400`</td><td>File is readable by owner only</td></tr><tr><td>`0664`</td><td>File is readable and writable by owner and members of the owner's group. File is readable by any user.</td></tr><tr><td>`0644`</td><td>File is readable and writable by owner. File is readable by members of the owner's group and any user.</td></tr></table> Default: `0640`, which makes the file readable and writable by the file owner and readable by members of the owner's group.

```
<AMPSConfig>
   ...
   <MiniDumpDirectory>/var/tmp</MiniDumpDirectory>
   <MiniDumpFileMask>0644</MiniDumpFileMask>
   ...
</AMPSConfig>
```

Example 3.3. MiniDump Configuration Example

3.4. Configuration Validation

The configuration validation option can be used to enable or disable the validation checking performed by AMPS on the initialization of each instance. Disabling the configuration validation can cause AMPS to start in an invalid state or not properly log warnings or errors in the configuration file.

Configuration validation should only be disabled during testing or debugging. We strongly recommend against disabling configuration validation in a production or development environment.

Table 3.6. Config Validation Parameters

Element	Description
`ConfigValidation`	Setting this to `disabled` will turn off AMPS configuration validation. The default is `enabled`, ensuring that the current AMPS configuration meets valid parameter ranges and data types. When this option is set to `disabled`, AMPS may start with an inconsistent or invalid configuration, which may have unpredictable effects.

```
<AMPSConfig>
  <ConfigValidation>enabled</ConfigValidation>
</AMPSConfig>
```

Example 3.4. Configuration Validation Example

3.5. Tuning

The `Tuning` section of the configuration file sets instance-level parameters for tuning the performance of AMPS. In many cases, AMPS self-tunes to take advantage of the hardware and environment. However, explicitly setting tuning parameters is sometimes necessary in cases where an AMPS instance cannot determine the best value. For example, if multiple AMPS servers are running on the same system, 60East recommends disabling NUMA.

Use the `Tuning` element with care. Options in the `Tuning` element can affect AMPS performance, and the behavior of `Tuning` options may be version-specific.

Table 3.7. Tuning Parameters

Element	Description
`NUMA/Enabled`	Setting this to `disabled` will turn off AMPS NUMA tuning. The default is `enabled`, which affinitizes certain AMPS threads to specific processors. The default value of `enabled` produces significantly better performance when a single instance of AMPS is running on a given system. However, if multiple instances of AMPS are running on the same system, setting this value to `disabled` for all of the instances on the system can reduce contention among the instances and produce better overall performance. Default: `enabled`

```
<AMPSConfig>
    <Tuning>
       <NUMA>
          <Enabled>enabled</Enabled>
       </NUMA>
    </Tuning>
</AMPSConfig>
```

Example 3.5. Tuning Example

3.6. Externals

The AMPS server depends on external libraries for some functionality. The Externals configuration item allows you to control the exact shared object loaded for some of these external libraries, particularly those related to security.

Table 3.8. Externals Parameters

Element	Description
`SSL/Library`	The path and shared object name of the SSL library to use for this instance. AMPS requires an SSL library that is compatible with OpenSSL 1.0.2 or later. By default, AMPS specifies the object name, and uses the standard shared object loading mechanism to resolve the object name. With this configuration option, you can direct AMPS to load a specific shared object. Default: `libopenssl.so`

Element	Description
`Crypto/Library`	The path and shared object name of the cryptography library. By default, AMPS specifies the object name, and uses the standard shared object loading mechanism to resolve the object name. With this configuration option, you can direct AMPS to load a specific shared object. Default: `libcrypto.so`
`Curl/Library`	The path and shared object name of the libcurl shared object. By default, AMPS specifies the object name and loads the version of libcurl included with the AMPS distribution as necessary. With this configuration option, you can direct AMPS to load a specific shared object. Default: `libcurl.so`

```
<AMPSConfig>
    <Externals>
       <SSL>
          <Library>/opt/audited/libopenssl.so</Library>
       </SSL>
       <Crypto>
          <Library>/opt/audited/libcrypto.so</Library>
       </Crypto>
       <Curl>
          <Library>/opt/resolver/lib/libcurl.so</Library>
       </Curl>
    </Externals>
</AMPSConfig>
```

Example 3.6. Externals Example

Chapter 4. Admin Server and Statistics

The `Admin` tag is used to control the behavior of the administration server and statistics collection for the instance.

Table 4.1. Admin Parameters

Element	Description
`InetAddr`	Defines a port for the embedded HTTP admin server, which can then be accessed via a browser. This element can also specify an IP address, in which case the HTTP server listens only on that address. If no IP address is specified, the HTTP server listens on all available addresses. There is no default for this parameter. If this parameter is not provided, AMPS does not provide an HTTP admin server, but will continue to collect statistics.
`FileName`	Location for storing the statistics information reported by the Admin Server. default: `:memory:` When the FileName is set to the default, the statistics database is maintained in memory.
`Interval`	The refresh interval for the Admin Server to update gathered statistics. default: `10s` minimum: `1s`
`Authentication`	The authentication to use for the Administrative interface. This is an `Authentication` element, as described in Chapter 12.
`Entitlement`	The entitlement to use for the Administrative interface. This is an `Entitlement` element, as described in Chapter 13.

```
<Admin>
  <InetAddr>localhost:9090</InetAddr>
  <FileName>stats.db</FileName>
  <Interval>20s</Interval>
</Admin>
```

Example 4.1. Admin Example

Chapter 5. Modules

The `Modules` section of the AMPS configuration file is used to load, configure and define any plug-in modules used for this installation of AMPS. AMPS supports a wide variety of plug-in modules, as described in the *Extending AMPS Guide*.

The following steps are required to use a plug-in module:

1. Load the module and declare the name of the module.

2. Define the AMPS object that the module contains and give the object a name and pass any required options.

3. Use the module in a specific context.

For many modules, such as `Authentication` and `Entitlement` modules, steps 2 and 3 are performed at the same time. Steps 2 and 3 above are separate when a module must have the same definition across mutliple contexts (for example, a `MessageType` which may be used in a Transport, a SOW, a View, and replicated to other instances).

The available features of a `Module` are listed in Table 5.1.

Table 5.1. Module Parameters

Element	Description
`Name`	A plain text name for the module. This will be used as a reference when the module is used elsewhere in the AMPS configuration, and is also the name that AMPS will use for logging messages related to the module.
`Library`	The shared object file that contains the compiled module. This must contain a path to the file. When using relative paths, those paths are evaluated relative to the current working directory of the AMPS process. For example to load a file from the current working directory, you must specify the directory (for example, `./my_awesome_module.so`). AMPS automatically searches the `lib` directory of the AMPS distribution for shared objects. If you install the shared object in the `lib` directory of the AMPS distribution, you can simply provide the filename of the shared object without using a path.
`Options`	A list of supported features for the implemented library. AMPS allows you to pass options to the module by specifying elements within the `Options` element. The exact options that the module requires, if any, are determined by the creator of the module.

Example 5.1 provides an example of an AMPS configuration using an authorization and entitlement plug-in module. In our example, a custom authentication module named `libauthenticate_customer001.so` has been written to manage the authentication portion of AMPS authentication. Similarly, a custom entitlements module has been written named `libentitlement_customer001.so` to manage the permissions and access of the authenticated user.

The first step is to define the global `Modules` section of the AMPS configuration, and then list the individual modules.

```
<AMPSConfig>
...
  <Modules>
    <Module>
      <Name>authentication1</Name>
      <Library>libauthenticate_customer001.so</Library>
      <Options>
        <LogLevel>info</LogLevel>
        <Mode>debugging</Mode>
      </Options>
    </Module>
    <Module>
      <Name>entitlement1</Name>
      <Library>libentitlement_customer001.so</Library>
      <Options>
        <LogLevel>error</LogLevel>
        <Mode>prod</Mode>
      </Options>
    </Module>
    ...
  </Modules>
...
</AMPSConfig>
```

Example 5.1. Sample global config of authentication and entitlements modules

We now have an authentication module and an entitlements module that we can reference elsewhere in the AMPS configuration file to enable authentication and/or entitlements for supported features. For example, we can create one type of `Authentication` module for the instance as a whole, and then create instances of a different type of Authentication and Entitlement modules for each `Transport`, to ensure that our `Transports` are properly enabling authentication and entitlements. In this example, the `Authentication` and `Entitlement` modules configured for an individual `Transport` are used for that transport, and the instance level modules are used as a default for transports that do not specify any `Authentication` or `Entitlement`.

This is accomplished via an entry similar to Example 5.2.

```
<AMPSConfig>
...
  <Authentication>
    <Module>amps-no-authorization</Module>
  </Authentication>
  <Entitlement>
    <Module>amps-no-authorization</Module>
  </Entitlement>
...
  <Transports>
    <Transport>
      <Name>fix-tcp-001</Name>
...
      <Authentication>
        <Module>authenticate_customer001</Module>
      </Authentication>
      <Entitlement>
        <Module>entitlement_customer001</Module>
      </Entitlement>
    </Transport>
    <Transport>
```

```
      <Name>fix-tcp-007</Name>
      ...
      <Authentication>
        <Module>authenticate_customer007</Module>
      </Authentication>
      <Entitlement>
        <Module>entitlement_customer007</Module>
      </Entitlement>
    </Transport>

    <Transport>
      <Name>json-tcp<Name>
      <!-- does not specify Authentication or
           entitlement, uses instance-level
           modules -->
      ...
    </Transport>

  </Transports>
...
</AMPSConfig>
```

Example 5.2. Example of security enabled transports

Example 5.2 shows how our `fix-tcp-001` transport is secured with the `authenticate_customer001` authentication module, and the `entitlement_customer001` entitlement module, which is defined in a global `Modules` section similar to the one listed in Example 5.1. Similarly, the `fix-tcp-007` transport is secured with the `authenticate_customer007` authentication module and the `entitlement_customer007` entitlement module. In contrast, the `json-tcp` transport does not define modules, and instead uses the authentication and entitlement modules specified at the instance level.

Chapter 6. Message Types

This tag defines the message types supported by the AMPS instance. A single AMPS instance can support multiple message types, as `MessageTypes` can contain multiple `MessageType` definitions.

`MessageType` definitions for `fix`, `nvfix`, `xml`, `json`, `bflat`, `bson`, and `binary` are automatically loaded by AMPS. You only need to define a new `MessageType` these if the settings for the message type need to be changed (for example, to create a custom FIX-based type that changes the `FieldSeparator` of the message).

AMPS loads the capability to use Google protocol buffer (`protobuf`) messages by default. To use protocol buffer messages, you configure one or more message types that use the `protobuf` module and load the `.proto` files that define the format of the messags you will be processing with AMPS.

AMPS also supports the ability to create a composite message type by combining a number of existing message types. Composite message types are defined using the `MessageType` configuration element.

Table 6.1. Message Type Parameters

Name	*Description*
`Name`	This element defines the name for the message type. The name is used to specify `MessageType` in other sections such as `Transport`, `TransactionLog` and the elements of the`SOW` section. By default, AMPS loads message types for `fix`, `nvfix`, `soapfix`, `json`, `bflat`, `bson`, `xml` and `binary`.
`Module`	The element specifies the name of the module that will be loaded for this message type. By default, AMPS loads the modules that implement the following message types: `fix`, `nvfix`, `soapfix`, `json`, `bflat`, `bson`, `xml`, `protobuf`, and `binary`. AMPS supports creating composite message types out of existing message types using the `composite-global` and `composite-local` modules, which are loaded by default.
`AMPSVersionCompliance`	Sets the version compatibility for FIX messages that AMPS sends to the `/AMPS/SOWStats` topic. AMPS accepts three values for this option: • `2` creates messages that use the FIX field tags used by AMPS 2.X versions. • `4` creates messages that use the default FIX field tags. With this version, FIX messages use different field numbering for `/AMPS/SOWStats` and `/AMPS/ClientStatus` messages. • `5` creates messages that use a unified set of FIX tags. When this option is set to 5, AMPS uses consistent field numbering between `/AMPS/SOWStats` and `/AMPS/ClientStatus` messages. Default: `4`. For compatibility with the largest number of existing installations, this parameter defaults to `4`.

Name	*Description*
	For message types other than FIX, there is no difference between 4 and 5. These message types were not supported in AMPS 2.X: AMPS provides reasonable values for these message types when this value is set to 2, but there is no backward compatibilty to enforce. For most cases, you can leave this option set to the default. If you are using a system that requires consistent FIX tags across messages, set this parameter to 5. If you are using an existing system that expects AMPS 2.X tags, set this parameter to 2.
`Options`	Options to pass to a custom message type module. AMPS does not specify the format or type of the elements within an `Options` element. AMPS simply parses the XML and then sends the XML to the module. If you are configuring a custom message type, see the documentation for that message type module for details.
`FieldSeparator`	*Option*: Applies to `fix` and `nvfix` message types. Sequence of characters used to separate field items in a FIX message. Note: this field is the ASCII value of the char sequence.
`HeaderSeparator`	*Option*: Applies to `fix` and `nvfix` message types. Sequence of characters used to separate the header from the body in a FIX message. Note: this field is the ASCII value of the char sequence.
`MessageSeparator`	*Option*: Applies to `fix` and `nvfix` message types. Sequence of characters used to separate message items in the body in a FIX message. Note: this field is the ASCII value of the char sequence.
`EarlyTerminationOptimization`	*Option*: Applies to the `json` message type. By default, AMPS includes a optimization to allow the server to to only partially parse JSON messages. This may result in unexpected behavior for some messages. For example, given a message such as `{ "code" : 1, "data" : "some data", "code" : 2 }`, AMPS will report the value of `code` as `1` when this optimization is active. To ensure consistent results, in this mode AMPS always reports the first value for a field even when AMPS fully parses the message. When set to `false`, the optimization is disabled. AMPS will fully parse all JSON messages and report the last value for a field. For the message above, AMPS would report the value of `code` as `2`. Default: `true`
`Type`	*Required*: Applies to message types that use the `protobuf` module. The name of the type within the `.proto` file to use for this message type. The name must be namespace-qualified (for example, `MyNamespace.Message` would load the type `Message` within the namespace `MyNamespace`).

Name	*Description*
	Obsolete usage A previous meaning of this element was made obsolete in AMPS 4.0 and later versions. That usage was replaced by the `Module` directive.
`MessageType`	*Required*: Applies to message types that use the `composite-local` or `composite-global` modules. For composite message types, the `MessageType` definition must contain one or more message type declarations that specify the types that the composite message type contains. See the *AMPS User Guide* for more information on composite message types.

`ProtoPath`	*Required*: Applies to message types that use the `protobuf` module.
	
	The path in which to search for `.proto` files. The content of this element has the following syntax:
	
	`*alias* ; *full-path*`
	
	The alias provides a short identifier to use when searching for `.proto` files. The full path is the path that is substituted for that identifier.
	
	A configuration may omit the alias, and simply provide the path. For example, to use the path /mnt/repository/protodefs when no alias is provided, you would declare a `ProtoPath` of:
	
	`;/mnt/repository/protodefs`
	
	The following ProtoPath declaration sets `proto-archive` as an alias for `/mnt/shared/protofiles`.
	
	`proto-archive;/mnt/shared/protofiles`
	
	AMPS uses the aliases provided in this configuration item when processing `import` statements within the loaded `.proto` files, with the empty alias used for import statements that do not specify a path alias. For example, given the definitions above, this import statement:
	
	`import "proto-archive/AType.proto";`
	
	will load for the file at `/mnt/shared/protofiles/AType.proto`, while the import statement:
	
	`import "MyFavoriteType.proto";`
	
	will load the file at `/mnt/repository/protodefs/MyFavoriteType.proto`.
	
	If no `ProtoPath` declaration sets an empty alias, all imports processed by AMPS must use one of the aliases set for the instance, or AMPS will fail to find the specified file.

Name	*Description*
	Unless your existing definitions use an aliasing scheme, it is most convenient to set the empty alias. You may specify any number of `ProtoPath` declarations.
`ProtoFile`	*Required*: Applies to message types that use Google protocol buffers. The name of the `.proto` file to use for this message type. To use an alias, prefix the name of the file with the alias. For example, if your ProtoPath declarations have created the `proto-archive` alias for the directory in which your `.proto` files are stored, you could use the following to use the `my-messages.proto` file within that directory. `proto-archive/my-messages.proto`

```
<MessageTypes>
  <!-- Define a FIX-based message type with custom separators -->
  <MessageType>
    <Name>fix-custom</Name>
    <Module>fix</Module>
    <!-- The following are FIX specific options -->
    <FieldSeparator>1</FieldSeparator>
    <HeaderSeparator>2</HeaderSeparator>
    <MessageSeparator>5</MessageSeparator>
  </MessageType>

  <!-- Define a message type for a custom
       payload. 'type-module' must be the
       Name of a Module specified in the
       configuration. -->
  <MessageType>
      <Name>custom-payload</Name>
      <Module>type-module</Module>
  </MessageType>

  <!-- Define a composite message type
       that combines a json message and
       a custom-payload message. -->

  <MessageType>
      <Name>custom-composite</Name>
      <Module>composite-local</Module>
      <MessageType>json</MessageType>
      <MessageType>custom-payload</MessageType>
  </MessageType>

</MessageTypes>
```

Example 6.1. Message Types Example

Chapter 7. Transports

The `Transports` element configures how AMPS communicates with publishers and subscribers, as well as how AMPS accepts connections for replication. The Transports element is a container for one or more `Transport` elements. Each `Transport` is a combination of a network transport, an AMPS header protocol, and a message type.

A `Transport` also specifies the `Authentication` used to validate the users that connect, and the `Entitlement` used to enforce permissions for users that connect over that transport.

AMPS supports a variety of network transports, header protocols and message formats for communication between publishers and subscribers. This section describes how to configure a `Transport`.

Table 7.1. Transport Parameters

Element	Description
`Name`	The name to use for this Transport. This name appears in the AMPS log for messages related to the transport.
`Protocol`	This element defines the protocol to use for sending and receiving messages. The protocol is typically `amps`, the name of a specific protocol for interoperability with another system or a legacy application, or the name of a custom protocol module specified in the `Modules` element.

AMPS provides support for the following protocols:

Table 7.2. Protocols

Protocol Name	Description
`amps`	Standard AMPS messaging, using compact headers in JSON format. AMPS accepts `json` as a synonym for `amps` in a protocol declaration.
`fix-session`	FIX session protocol, for use with systems that publish FIX messages using this format.
`websocket`	Websocket protocol, using JSON format headers.
Legacy protocols	
`fix`	Standard AMPS messaging, using headers in FIX format.
`nvfix`	Standard AMPS messaging, using headers in NVFIX format.
`soap`	Standard AMPS messaging, using headers in SOAP format.
`xml`	Standard AMPS messaging, using headers in XML format.

60East recommends using the `amps` protocol for general purpose AMPS messaging. When your application uses the the FIX session layer or Websockets, use those protocols.

Element	Description
	Older versions of AMPS used message headers in the same format as the message type: if your instance supports applications that expect to use a specific message type protocol, use that protocol in your `Transport` configuration.
`Type`	The type of Transport. Valid values include: `tcp`, `tcps`, `amps-replication`, `amps-replication-secure`, `amps-unix` `tcp` is the standard TCP transport. `tcps` is secure TCP transport: this transport type uses SSL and requires a certificate and private key to be set. `amps-replication` is for inbound replication connections. Notice that AMPS replication does not use the same transport type as other applications. `amps-replication-secure` is for inbound replication connections that use SSL. Notice that AMPS replication does not use the same transport type as other applications. `amps-unix` is for incoming client connections over Unix domain sockets. This transport type requires a `FileName`, which is the location on the file system where the Unix domain socket will be created.
`InetAddr`	The port on which AMPS will listen for this transport. This element can also specify an IP address, in which case AMPS listens only on that address. If no IP address is specified, AMPS listens on all available addresses. This element is not required for transports of the `amps-unix` `Type`, but is required for all other `Type` values.
`MessageType`	Restricts a transport to a single message type. When provided, AMPS assumes that all connections to this transport use the specified `MessageType`. If a different `MessageType` is provided in the connection string, AMPS refuses the connection. If no `MessageType` is provided in the connection string, AMPS accepts the connection and assumes that the connection uses the specified `MessageType`. When the `Transport` `Type` is `amps-replication`, this element is ignored and the `Transport` accepts all message types configured in the instance. When present, this is a reference to the name of a specific message type defined in the `MessageTypes` section or one of the message types that AMPS loads by default. In this release, AMPS loads the following message types by default: `fix`, `nvfix`, `xml`, `json`, `bson`, `bflat` and `binary`. Composite message types, and message types based on Google Protocol Buffers, must be defined in the `MessageTypes` element before they can be used. A `Transport` that uses the `amps` `Protocol` defaults to accepting all message types defined by the instance, and does not require setting a `MessageType` element. When the `Transport` does not specify a `Mes-`

Element	Description
	sageType, the connecting client must declare the message type it will use when logging on. A Transport that uses the amps protocol can specify a single MessageType to accept by including this element. When a single MessageType is specified, AMPS does not require that the message type is specfied by the client. *Important*: A Transport that uses one of the legacy Protocol values (fix, nvfix, or xml) ***must*** specify a MessageType. Default: When the Protocol is amps, defaults to accepting all message types defined by the instance. When the Protocol is one of the legacy values, there is no default and a MessageType must be provided.
InitialState	Defines whether, when AMPS starts, the transport is enabled or disabled. When the transport is disabled, AMPS does not listen for or accept connections on the transport. When InitialState is disabled, the transport must be explicitly enabled after startup (for example, through an action or the administrative console) for AMPS to listen for and accept connections on the transport. This configuration option can be useful for defining a transport that is only available when certain conditions are true: for example, an instance might start with the connection used by clients disabled, and let an external monitoring system enable the connection during business hours and disable the connection outside of business hours. Default: enabled
ReuseAddr	Permits an AMPS instance to use a socket that is in a WAIT state. This can occur when AMPS has been restarted using the same InetAddr and the previous instance did not fully close the port. This option is not used by transports of the amps-unix Type. Valid values: true or false Default: false
Entitlement	Specifies the entitlement module to use for this transport. If no entitlement module is provided, the transport uses the default entitlement module for the instance. This element must contain a Module element with the Name of an entitlement module. If the module requires options, those options are provided in an Options element within the Entitlement element. Default: The module specified in the Entitlement for the instance (defaults to amps-default-entitlement-module if not provided)
Authentication	Specifies the authentication module to use for this transport. If no authentication module is provided, the transport uses the authentication module for the instance. This element must contain a Module element with the Name of an authentication module. If the module requires options, those options are provided in an Options element within the Authentication element.

Element	Description
	Default: The module specified in the `Authentication` element for the instance (defaults to `amps-default-authentication-module` if not provided)
`MessageMemoryLimit`	The total amount of memory to allocate to messages before offlining clients. Default: The setting configured at the instance level. If this option is not specifically set at the instance level, the instance defaults to 10% of total host memory or 10% of the amount of host memory AMPS is allowed to consume (as reported by `ulimit -m`), whichever is *lowest*.
`MessageDiskLimit`	The total amount of disk space to allocate to messages before disconnecting clients. Default: The setting configured at the instance level. If this option is not specifically set at the instance level, the instance defaults to 1GB or the amount specified in the MessageMemoryLimit, whichever is *highest*.
`MessageDiskPath`	The path to use to write offline files. Default: `/var/tmp`, or the setting configured at the instance level
`ClientMessageAgeLimit`	The maximum amount of time for the client to lag behind. If a message for the client has been held longer than this time, the client will be disconnected. This parameter is an AMPS time interval (for example, `30s` for 30 seconds, or `1h` for 1 hour). Default: No age limit, or the setting configured at the instance level
`ClientMaxCapacity`	The amount of available capacity a single client can consume. Before a client is offlined, this limit applies to the `MessageMemoryLimit`. After a client is offlined, this limit applies to the `MessageDiskLimit`. This parameter is a percentage of the total. Default: `100%` (no effective limit), or the setting configured at the instance level

Starting with 5.1, AMPS supports encrypting client connections using the SSL (Secure Sockets Layer) network protocol. The following parameters apply to transports that use SSL. While AMPS performs additional configuration validation if the transport is configured with a `Type` of `tcps`, if a `Certificate` and `PrivateKey` are specified for a `Transport` of `tcp` type, AMPS will use SSL for that `Transport`.

Table 7.3. SSL Transport Parameters

Element	Description
`Certificate`	The certificate file to use for the server. Default: There is no default for this option.
`PrivateKey`	The private key to use for the server. Default: There is no default for this option.
`Ciphers`	The cipher list to use for this transport. The cipher list is passed to the OpenSSL implementation without being interpreted by the AMPS server. Default: There is no default for this option. For OpenSSL, details on the format of the cipher list are available at https://www.openssl.org/docs/manmaster/apps/ciphers.html.

For protocols of Type amps-unix, AMPS requires the following additional configuration option:

Table 7.4. Websocket Transport Parameters

Element	Description
FileName	Specifies the location on the filesystem where the Unix-domain socket will be created. This location is the path that will be provided to clients that need to connect using this transport. This element is required for transports of type amps-unix. There is no default.

For protocols of Type websocket, AMPS supports the following additional configuration option:

Table 7.5. Websocket Transport Parameters

Element	Description
PerMessageDeflate	Controls per-message deflation for websocket connections. When this value is disabled, AMPS does not perform per-message deflation. If this option is not present, or is set to any other value, AMPS performs per-message deflation. Per-message deflation is normally negotiated between an application and AMPS during the opening handshake of a websocket connection. 60East recommends leaving this option enabled unless you have a specific reason for disabling it. Default: enabled

```
<Transports>

  <!-- fix messages using TCP -->
  <Transport>
    <Name>fix-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9004</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>fix</MessageType>
  </Transport>

  <!-- nvfix messages using TCP -->
  <Transport>
    <Name>nvfix-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9005</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>nvfix</MessageType>
  </Transport>

  <!-- xml messages using TCP -->
  <Transport>
    <Name>soap-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9006</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>xml</MessageType>
  </Transport>

</Transports>
```

Example 7.1. Transports Example

```
<AMPSConfig>

 ...


 <MessageMemoryLimit>10GB</MessageMemoryLimit>
 <MessageDiskPath>/mnt/fastio/AMPS/offline</MessageDiskPath>
 <ClientMessageAgeLimit>30s</ClientMessageAgeLimit>

...
  <Transports>

  <!-- This transport shares the 10GB MessageMemoryLimit
       defined for the instance. -->
  <Transport>
    <Name>regular-json-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9007</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>json</MessageType>
  </Transport>

  <!-- This transport shares the 10GB MessageMemoryLimit
       defined for the instance. -->
  <Transport>
    <Name>regular-bson-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9010</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>bson</MessageType>
    <!-- However, this transport does not allow
         clients to fall as far behind as the
         instance-level setting -->
    <ClientMessageAgeLimit>15s</ClientMessageAgeLimit>
  </Transport>


  <!-- This transport has a separate 35GB MessageMemoryLimit
       and a 70GB MessageDiskLimit. It uses the instance-wide
       30s parameter for the ClientMessageAgeLimit -->
  <Transport>
    <Name>highpri-json-tcp</Name>
    <Type>tcp</Type>
    <InetAddr>9995</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <MessageType>json</MessageType>
    <MessageMemoryLimit>35GB</MessageMemoryLimit>
    <MessageDiskLimit>70GB</MessageDiskLimit>
  </Transport>

  </Transports>

</AMPSConfig>
```

Example 7.2. Transports Example with Resource Management

```
<AMPSConfig>

  <Transports>

  <Transport>
    <Name>ssl-all-message-types</Name>
    <Type>tcps</Type>
    <InetAddr>9007</InetAddr>
    <ReuseAddr>true</ReuseAddr>
    <Certificate>${AMPS_INSTALL}/cert.pem</Certificate>
    <PrivateKey>${AMPS_INSTALL}/key.pem</PrivateKey>
    <Ciphers>HIGH:!aNULL</Ciphers>
  </Transport>

  </Transports>

</AMPSConfig>
```

Example 7.3. SSL Transport Example

Chapter 8. Logging

AMPS supports several different types of log formats, and multiple targets can be defined simultaneously.

Table 8.1. Logging Parameters

Element	Description
`Protocol`	Define the logging target protocol Valid values: `stdout, stderr, file, gzip, syslog`
`FileName`	File to log to. If `RotationThreshold` is specified, then `-%n` is added to the file. If the protocol is gzip, then `.gz` is added to the file name Default: `$PWD/%Y-%m-%dT%H%M%S.log`
`RotationThreshold`	Log size at which log rotation will occur. See Table 1.3 for details on specifying file size.
`Level`	Defines a lower bound (inclusive) log level for logging. All log messages at the specified level and up are logged. Valid values: `none`, `trace`, `debug`, `stats`, `info`, `warning`, `error`, `critical`, `emergency`
`Levels`	A comma separated list of specific log levels. Only log messages at the specified levels will be logged. This element can be used with the `Level` element. In that case, the AMPS will log all messages at `Level` and above, and in addition, will log errors at the levels specified by `Levels`. Valid values: `developer`, `trace`, `debug`, `stats`, `info`, `warning`, `error`, `critical`, `emergency`, `none`
`IncludeErrors`	Additional errors that should be included when logging. If an error appears in this element, it will be logged regardless of the level of the error. This element accepts a comma-delimited list of error numbers. You can also provide a regular expression that matches a set of errors, such as `12-.*`
`ExcludeErrors`	Errors that should be excluded when logging. If an error appears in this element, it will not be logged regardless of the level of the error. If the same error appears in both `IncludeErrors` and `ExcludeErrors`, `ExcludeErrors` takes precedence and the error will not be logged. This element accepts a comma-delimited list of error numbers. You can also provide a regular expression that matches a set of errors, such as `12-.*`
`Ident`	Syslog identifier for the AMPS instance. Default: AMPS Instance Name
`Options`	A comma separated list of syslog options. If using `syslog`, 60East recommends using `LOG_CONS`, `LOG_NDELAY`, and `LOG_PID`. AMPS uses the standard options to syslog, as described in the syslog man page.
`Facility`	Syslog facility to use.

```
<Logging>
  <!-- trace messages to files in /var/tmp/ -->
  <Target>
    <Protocol>file</Protocol>
    <FileName>
      /var/tmp/amps/logs/%Y%m%d%H%M%S-%n.log
    </FileName>
    <RotationThreshold>2G</RotationThreshold>
    <Level>trace</Level>
    <Levels>critical</Levels>
  </Target>
  <!-- Record critical messages to the system logger. -->
  <Target>
    <Protocol>syslog</Protocol>
    <Level>critical</Level>
    <Ident>amps_dma</Ident>
    <Options>LOG_CONS,LOG_NDELAY,LOG_PID</Options>
    <Facility>LOG_USER</Facility>
  </Target>
  <!-- record only the AMPS initialization
       message -->
  <Target>
    <Protocol>file</Protocol>
    <FileName>/var/tmp/amps/logs/initMessage</FileName>
    <IncludeErrors>00-0015</IncludeErrors>
</Logging>
```

Example 8.1. Logging Example

Chapter 9. State-of-the-World (SOW) Features

The `SOW` section of the configuration file holds the configuration for the AMPS State of the World.

Table 9.1. SOW section configuration elements

Element	Description
`Topic`	Specifies that AMPS will record distinct messages for this topic in the State of the World. SOW topic definitions are used directly as a last-value cache, and are required for many of the advanced messaging features in AMPS such as out-of-focus notifications and delta messaging. SOW topic definitions can also be used as the underlying topics for views, aggregates, and conflated topics. SOW `Topic` configuration is described in Section 9.1.
`Queue`	Defines a message queue. Rather than delivering each message to all matching subscriptions, message queues provide features to help ensure that each message is delivered to and processed by a single subscriber. See the User Guide for a full description of the functionality of message queues. Queue configuration is described in Section 9.2.
`View`	Defines a View over one or more SOW topics, conflated topics, or other views. A view can perform aggregation, can JOIN multiple topics together. A view can be based on a SOW topic of one message type and project results of a different message type. View configuration is described in Section 9.3.
`Conflated-Topic`	Defines a copy of a SOW topic or view that receives current value updates at a specified interval, conflating any changes to values that occur between the scheduled updates. ConflatedTopic configuration is described in Section 9.4.

The elements within the SOW section are described in detail in the following sections.

9.1. SOW/Topic

Creating a `Topic` element in the the SOW configuration section declares that AMPS will maintain state of the world for that topic. When a topic is recorded in the State of the World (SOW), AMPS maintains the most recent value of each distinct message in that topic. In this respect, the SOW for a topic is similar to a table in a relational database.

Notice that AMPS does not require topics to be predeclared. Creating this configuration item is declaring that the State of the World will be maintained for this topic: the configuration *does not* define a topic. Creating this configuration item is not required to publish messages to the topic.

There are several sets of options required when a topic is added to the State of the World. *General* options define the topic name and the overall behavior of the State of the World for this topic. *Key specification* options define how AMPS identifies unique messages within the topic (much like declaring a primary key for a table in a relational database). *Growth specification* options define how AMPS grows storage for the State of the World for this topic as messages are added.

Table 9.2 contains a listing of the parameters for a `Topic` section in the `SOW` section of an AMPS configuration file. AMPS accepts `TopicDefinition` as a synonym for `Topic`, since this was the element name used in versions before 5.0.

This section presents brief descriptions of the options for recording the State of the World for a topic. For full information on the State of the World, see the *AMPS User Guide*.

Table 9.2. SOW/Topic General Options

Element	Description
`FileName`	The file where the State of the World (SOW) data will be stored. This element is required for SOW topics with a `Durability` of `persistent` (the default) because those topics are persisted to the filesystem. This is not required for SOW topics with a durability of `transient`.
`MessageType`	Type of messages to be stored. To use AMPS generated SOW keys, the message type specified must support content filtering so that AMPS can determine the SOW key for the message. All of the default message types, except `binary`, support content filtering. Since the `binary` message type does not support content filtering, that type can only be used for a SOW when publishers use explict keys. See the "Message Types" chapter in the *AMPS User Guide* for a discussion of the message types that AMPS loads by default. Some message types (such as Google Protocol Buffers) require additional configuration, and must be configured before using the message type in a SOW topic.
`Name`	The name of the SOW topic - all unique messages on this topic will be stored in a topic-specific SOW database. Every SOW requires a method of determining which messages are unique. Several methods are provided within AMPS. See the AMPS User Guide for a discussion on SOW keys, and Table 9.3 for relevant configuration items. If no `Name` is provided, AMPS accepts `Topic` as a synonym for `Name` to provide compatibility with versions of AMPS previous to 5.0.
`HashIndex`	AMPS provides the ability to do fast lookup for SOW records based on specific fields. When one or more `HashIndex` elements are provided, AMPS creates a hash index for the fields specified in the element. These indexes are created on startup, and are kept up to date as records are added, removed, and updated. The `HashIndex` element contains a `Key` element for each field in the hash index. AMPS uses a hash index when a query uses a exact matching for all of the fields in the index. AMPS does not use hash indexes for range queries or regular expressions. AMPS automatically creates a hash index for the set of fields specified in the set of `Key` fields for the SOW, if those fields are specified.
`Index`	AMPS supports the ability to precreate memo indexes for specific fields using the `Index` configuration option. When one or more `Index` elements are provided, AMPS creates memo indexes for any field specified in an `Index` element on startup, before a query that uses that field runs. Otherwise, AMPS indexes each field the first time a query uses the field. Adding one or more `Index` configurations to a `SOW/Topic` can improve retrieval performance the first time a query that contains the indexed fields runs for large SOW topics.
`RecoveryPoint`	For SOW topics that are covered by the transaction log, the point from which to recover the SOW if the SOW file is removed, or if the SOW topic has `transient` duration. This configuration item allows two values:

Element	Description
	• `epoch` recovers the SOW from the beginning of the transaction log • `now` recovers the SOW from the current point in the transaction log Defaults to `epoch`.
`Expiration`	Time for how long a record should live in the SOW database for this topic. The expiration time is stored on each message, so changing the expiration time in the configuration file will not affect the expiration of messages currently in the SOW. AMPS accepts interval values for the Expiration, using the interval format described in the AMPS Configuration Guide section on units, or one of the following special values: • A value of `disabled` specifies that AMPS will not process SOW expiration for this topic, regardless of any expiration value set on the message. In this case, AMPS saves the expiration for the message, but does not process it. The value must be set to `disabled` (the default) if `History` is enabled for this topic. • A value of `enabled` specifies that AMPS will process SOW expiration for this topic, with no expiration set by default. Instead, AMPS uses the value set on the individual messages (with no expiration set for messages that do not contain an expiration value). Default: `disabled` (never expire)
`Durability`	Defines the data durability of a SOW topic. SOW databases listed as `persistent` are stored to the file system, and retain their data across instance restarts. Those listed as `transient` are not persisted to the file system, and are reset each time the AMPS instance restarts. Default: `persistent` Valid values: `persistent` or `transient` Synonyms: `Duration` is also accepted for this parameter for backward compatibility with configuration prior to 4.0.0.1
`History`	Enable historical query for this SOW. This element contains a `Window` and `Granularity` element. When the `History` element is present, historical query is enabled for this sow. Otherwise, AMPS does not enable historical query and does not store the historical state of the SOW. `Expiration` must be `disabled` when `History` is enabled.
`Window`	For a historical SOW, the length of time to store history. For example, when the value is `1w`, AMPS will store one week of history for this SOW. Used within the `History` element. Default: By default, AMPS does not expire historical SOW data.
`Granularity`	For a historical SOW, the granularity of the history to store. For many applications, it is not necessary for AMPS to store all of the updates to the SOW. This parameter sets the resolution at which AMPS will save the state of a message.

Element	Description
	For example, when you set a granularity of `1m`, AMPS will save the state of the message no more frequently than once per minute, even when the state of the message is updated several times a minute. Used within the `History` element.
`Preprocessing`	When present, specifies the message enrichment to be performed *before* AMPS determines the SOW key for the message. The `Preprocessing` element must contain one or more `Field` elements that specify the enrichment to perform.
`Enrichment`	When present, specifies the message enrichment to be performed *after* AMPS determines the SOW key for the message. The `Enrichment` element must contain one or more `Field` elements that specify the enrichment to perform.

Each SOW topic must define how AMPS will determine which messages are unique. An application can either have AMPS determine the key by specifying one or more `Key` fields, provide a SOW key with the `publish` command each time a message is published to AMPS. AMPS also provides the ability to provide a custom SowKey generator with a plugin module.

See the appropriate section in the *AMPS User Guide* for a full discussion. The following table lists the available configuration items:

Table 9.3. SOW/Topic Key Specification Options

Element	Description
`Key`	Specifies an XPath within each message that AMPS will use to generate a SOW key, which determines whether a message is unique. This element can be specified multiple times to create a composite key from the combined value of the specified `Key` elements. When one or more `Key` elements is specified for the SOW, AMPS generates the SOW key for each message. When no `Key` fields are specified and no `KeyGenerator` is specified, publishers must explicitly provide the SOW key for each message when the message is published. 60East recommends configuring a `Key` and having AMPS generate the SOW key for a message unless your application has specific needs that make this impractical. AMPS automatically creates a hash index for the set of fields specified in the `Key` elements. There is no default for this element.
`KeyDomain`	The seed value for `SowKeys` used within the topic when AMPS generates the SOW key. The default is the topic name, but it can be changed to a string value to unify `SowKey` values between different topics. For example, if your application has a `ShippingAddress` SOW and a `CreditRating` SOW that both use `/customerID` as the SOW key, you can use a `KeyDomain` to ensure that the generated `SowKey` for a given `/customerId` is identical for both SOW topics. This does not affect how AMPS processes the SOW topics, but can make correlating information from different SOW topics easier in your application.

Element	Description
	This option can only be specified when one or more `Key` fields are specified. When a SOW key generator module is used, or the publisher must send a SOW key, this option is not valid. Default: the name of the SOW topic.
`KeyGenerator`	Specifies the SOW key generator module to use for this topic. When this configuration element is present, AMPS calls the specified module to generate a SOW key for each incoming message. Default: no SOW key generator module. When there is no SOW key generator module specified, AMPS uses the specified `Key` fields if the `Key` fields are provided. If no generator is specified and no `Key` fields are specified, AMPS requires publishers to set a SOW key on each message published. A KeyGenerator element contains the following elements:

Table 9.4. SOW/Topic/KeyGenerator Options

Option	Description
`Module`	The name of the module. This module must be loaded elsewhere in the configuration file.
`Options`	One or more XML elements. These elements are provided to the key generator module as options. The options provided depend on the key generator. The creator of the key generator module must document the options for that module.

The SOW topic configuration also specifies how the SOW file is allowed to grow. See the *AMPS User Guide* for details on SOW growth. The configuration options for controlling how the file is allocated and how the file grows are listed below:

Table 9.5. SOW/Topic Growth Specification Options

Element	Description
`SlabSize`	The size of each allocation for the SOW file, as a number of bytes. When AMPS needs more space for the SOW, it requests this amount of space from the operating system. This effectively sets the maximum message size that AMPS guarantees can be stored in the SOW. This size includes headers set by AMPS on the message. 60East recommends setting this value only if you will be storing messages larger than the default `SlabSize` or if performance or capacity testing indicates a need to tune SOW performance. If you plan to store messages larger than the default setting, 60East recommends a starting value of several times the maximum message size. For example, if your maximum message size is 2MB, a good starting point for `SlabSize` would be 8MB. If it becomes necessary to tune the `SlabSize`, see the *Best Practices* and *Capacity Planning* sections of the AMPS User Guide for a full discussion tuning the `SlabSize`.

Element	Description
	Default: `1MB`
`InitialSlabCount`	The number of SOW slabs that AMPS will allocate on startup. Default: `1` Maximum: `1024`
DEPRECATED: `RecordSize`	*This parameter is deprecated beginning in AMPS 5.0. Use the* `SlabSize` *parameter instead.* Size (in bytes) of a SOW record for this topic. Default: `512`
DEPRECATED: `InitialSize`	*This parameter is deprecated beginning in AMPS 5.0. Use the* `InitialSlabCount` *parameter instead.*Initial size (in records) of the SOW database file for this topic. Default: `2048`
DEPRECATED: `IncrementSize`	*This parameter is deprecated beginning in AMPS 5.0. Use the* `SlabSize` *parameter instead.* Number of records to expand the SOW database (for this topic) by when more space is required. Default: `1000`

An example of a SOW configuration looks like the following:

```
<SOW>

  <!-- Simple SOW topic definition -->
  <Topic>
    <Name>orders</Name>
    <Key>/orderId</Key>
    <MessageType>nvfix</MessageType>
    <FileName>./sow/%n.sow</FileName>
  </Topic>

  <!-- SOW with hash indexes -->
  <Topic>
    <Name>customers</Name>
    <Key>/customerId</Key>
    <MessageType>json</MessageType>
    <FileName>./sow/%n.sow</FileName>
    <HashIndex>
      <Key>/customerName</Key>
    </HashIndex>
    <HashIndex>
      <Key>/zipCode</Key>
      <Key>/customerType</Key
    </HashIndex>
  </Topic>

  <!-- Historical SOW -->
  <Topic>
    <Name>catalog</Name>
    <Key>/sku</Key>
    <MessageType>json</MessageType>
    <FileName>./sow/%n.sow</FileName>
    <History>
       <Window>7d</Window>
       <Granularity>15m</Granularity>
    </History>
  </Topic>

  <!-- SOW with enrichment

       Add a /fullName field constructed
       from /firstName and /lastName.
   -->
  <Topic>
      <Name>sales-reps</Name>
      <Key>/employeeId</Key>
      <MessageType>bflat</MessageType>
      <Enrichment>
         <Field>CONCAT(/firstName, " ", /lastName)
                AS /fullName</Field>
      </Enrichment>
  </Topic>
</SOW>
```

Example 9.1. SOW Topic Configuration

9.2. SOW/Queue and SOW/LocalQueue

This section lists configuration parameters for queues.

The `Queue` tag and the `LocalQueue` tag are used to configure message queues.

When an AMPS queue is defined with the `Queue` tag, the queue will be a distributed queue. To make a queue that is limited to the local instance, use the `LocalQueue` tag.

AMPS accepts `QueueDefinition` as a synonym for `Queue`.

Table 9.6. Queue configuration elements

Element	Description
`Name`	The name of the queue topic. This name is the name that consumers subscribe to. If no `Name` is provided, AMPS accepts `Topic` as a synonym for `Name` in the `Queue` definition.
`MessageType`	The message type of the queue.
`UnderlyingTopic`	A topic name or regular expression for the topic that contains the messages to capture in the queue. These topics must be recorded in a transaction log, and all must be of the same message type as the queue.

Element	Description
	If an `UnderlyingTopic` is not provided, the `UnderlyingTopic` defaults to the `Name` of the queue.
`DefaultPublishTarget`	The topic to publish to when an application publishes a message to the queue. For simplicity, AMPS allows applications to publish messages to the queue, and for those messages to be routed to one of the underlying topics. This element is required if the `UnderlyingTopic` contains regular expression characters. Otherwise, the `UnderlyingTopic` is a single topic and this element is optional and defaults to the `UnderlyingTopic`.
`LeasePeriod`	The amount of time that a subscriber has ownership of the message before the message is returned to the queue. For `at-least-once` delivery semantics, the consumer must process and acknowledge the message within this lease period, or the message may be provided to another subscriber. The `LeasePeriod` is measured from the time that AMPS sends the message to the subscriber. Set the `LeasePeriod` to account for round trip network latency as well as the expected processing time for the subscribers. Default: `infinite` (no expiration)
`Semantics`	The delivery semantics to use for this queue. There are two accepted values: • `at-least-once` With these semantics, you can guarantee that a message has been processed by at least one subscriber, as described in the introduction to Queues in the *AMPS User Guide*. With this value, a subscriber must explicitly remove the message from the queue once the message is processed. • `at-most-once` With these semantics, AMPS removes the message from the queue immediately when AMPS sends the message. This allows you to guarantee that no more than one subscriber will process the message. Default: `at-least-once`
`MaxBacklog`	The maximum number of outstanding, unacknowledged messages in the queue at any one time. This parameter allows you to set limits on the number of pending messages from the queue overall. When the queue reaches the MaxBacklog, no incoming messages are delivered from the queue until a message is removed from the queue (either by expiring, or being acknowledged by a client). This parameter allows you to avoid overwhelming clients during periods of heavy activity. Notice that this does not set a limit of any sort on the capacity of the queue. This parameter allows you to limit the

Element	Description
	number of messages that the queue will make available to subscribers at a given time, but does not restrict the capacity of the queue to track messages. Default: `infinite`
`MaxPerSubscriptionBacklog`	The maximum number of outstanding, unacknowledged messages in the queue for an individual subscription. This parameter allows you to avoid overwhelming a single subscriber during a period of heavy activity. Subscribers can declare the maximum number of messages that the subscription is prepared to lease at a given time. This maximum defaults to 1 when there is no maximum explicitly specified for a subscription. AMPS will lease the number specified in the subscription or the maximum set for the queue, whichever is lower. Notice that this does not set a limit of any sort on the capacity of the queue. This parameter allows you to limit the number of messages that the queue will make available for a single subscription at a given time, but does not restrict the capacity of the queue to track messages. Default: `1`
`Expiration`	The length of time a message can remain in the queue before AMPS considers the message undeliverable. Messages may expire while a subscriber has a lease on the message. AMPS does not send an additional notification in this case. Default: `infinite`
`Filter`	An AMPS `Filter` that is applied to the `UnderlyingTopic`. When a `Filter` is specified, only messages matching the `Filter` appear in the queue. By default, there is no filter and all messages from the UnderlyingTopic are presented in the queue.
`RecoveryPoint`	This option allows you to specify the point at which AMPS begins reviewing the transaction log to recover the state of the queue when AMPS restarts. By default, AMPS reviews the full log to determine the contents and state of the queue. The `RecoveryPoint` can be one of the following: • `epoch` - Recovery begins at the beginning of the transaction log • `creation` - Recovery begins at the time the queue was created

Element	Description
	• AMPS bookmark - When an AMPS bookmark is provided, AMPS starts recovery at the specified bookmark. • ISO-8601 timestamp - When a timestamp is provided, AMPS starts recovery at the specified timestamp. Default: `epoch`
`FairnessModel`	AMPS provides different methods to distribute messages across active subscriptions: • `fast` - AMPS delivers to the first subscription found that can process the message • `round-robin` - AMPS distributes to the next subscription found that can process the message • `proportional` - AMPS delivers to the subscription with the lowest ratio of active messages to available backlog Default: `proportional` for `at-least-once` queues, `round-robin` for `at-most-once` queues
`Leasing`	Ownership model for leased messages. AMPS supports the following models: • `strict` - AMPS allows a client to acknowledge (`sow_delete`) only messages that are leased to the client or currently unleased. If a client acknowledges a message leased to another client, there is no effect. • `sublet` - AMPS allows any client to acknowledge any message, regardless of whether another client has a lease on the message. Default: `sublet`

```
<!--
    Notice that the topics to use for
    the queue (ORDERS_.*) must be
    recorded in a transaction log.
-->

<SOW>
  <Queue>
    <Name>MQ</Name>
    <MessageType>json</MessageType>
    <UnderlyingTopic>ORDERS_.*</UnderlyingTopic>
    <DefaultPublishTarget>ORDERS_DIRECT</DefaultPublishTarget>
    <LeasePeriod>60s</LeasePeriod>
    <Expiration>1d</Expiration>
    <MaxBacklog>3</MaxBacklog>
  </Queue>
```

```
</SOW>
```

Example 9.2. Queue Example

9.3. SOW/View

Table 9.7 contains a listing of the parameters for a `View` section in the `SOW` section of an AMPS configuration file. For backward compatibility, AMPS accepts `ViewDefinition` as a synonym for `View`.

Table 9.7. SOW/View

Element	Definition
`MessageType`	The message type of the view. This does not need to be the same type as any of the topics in the aggregation, but does need to be a message type that supports views. The message type must be one of the message types configured for the instance. AMPS includes `fix`, `xml`, `nvfix`, `json`, `bson` and `bflat` message types with full support for views. You can also use any custom message type defined for the configuration file, provided that the message type supports views. Notice that the `binary` message type does not specify a fixed format for the message contents, so that message type cannot be used in a view. Other message types provided with AMPS may have limitations in their support for views. See the AMPS User Guide for the message type for a discussion of the limitations.
`Name`	Defines the topic name for this view. If no `Name` is provided, AMPS accepts `Topic` as a synonym for `Name` to provide compatibility with versions of AMPS previous to 5.0.
`UnderlyingTopic`	Defines the SOW topic or topics on which this view is based. This element can contain a single topic name, or any number of `Join` elements.
`Projection/Field`	Defines what the view will contain. This element can be specified multiple times to compose a complex view. Complex expressions that use aggregation functions and conditional branching can also be used.
`Grouping/Field`	Defines how the records in the underlying topic will be grouped. This is analogous to a SQL GROUP BY clause.
`KeyDomain`	The seed value for `SowKeys` used within this topic. The default is the topic name, but it can be changed to a string value to unify `SowKey` values between different topics.
`Join`	Within an `UnderlyingTopic`, each `Join` specifies two topics to join together to create the view, as well as the relationship between those topics. An `UnderlyingTopic` can have any number of `Join` specifications. For more information on `Join` specifications, see the *AMPS User Guide*.
`Conflation`	Defines whether AMPS will attempt to conflate updates to the view. This item accepts two values: • `none` - No conflation. AMPS fully processes every update to the view, in publication order.

Element	Definition
	• `inline` - Conflation is active. AMPS conflates updates to the same underlying record where possible. See the *AMPS User Guide* for details. Default: `none`
`Filter`	Defines a filter for the view. When provided, only messages from the underlying topic that match this filter will be included in the view. This option is only valid when the view uses a *single* `UnderlyingTopic`. When the view contains a `Join` specification, this option may not be used. Default: No filter, which includes all messages from the `UnderlyingTopic`.
`FileName`	File location to store view data. Unused in this version of AMPS.

```
<SOW>
  <!-- Single topic aggregation -->
  <Topic>
    <Topic>/ett/order</Topic>
    <MessageType>fix</MessageType>
    <Key>/orderId</Key>
  </Topic>
  <View>
    <FileName>./sow/%n.view.sow</FileName>
    <MessageType>nvfix</MessageType>
    <Topic>TOTAL_VALUE</Topic>
    <UnderlyingTopic>/ett/order</UnderlyingTopic>
    <Projection>
      <Field>/109</Field>
      <Field>SUM(/14 * /6) AS /71406</Field>
    </Projection>
    <Grouping>
      <Field>/109</Field>
    </Grouping>
  </View>

  <!-- Single topic aggregation with filter -->
  <Topic>
     <Name>orders</Name>
     <MessageType>json</MessageType>
     <Key>/orderId</Key>
     <FileName>./sow/%n.sow</FileName>
  </Topic>
  <View>
     <Name>CompleteByRegion</Name>
     <UnderlyingTopic>orders</UnderlyingTopic>
     <MessageType>json</MessageType>
     <Projection>
       <Field>COUNT(/orderId) AS /completedOrders</Field>
       <Field>/region AS /region</Field>
     </Projection>
     <Grouping>
```

```
        <Field>/region</Field>
     </Grouping>
     <Filter>/status = 'complete'</Filter>
   </View>

  <!-- Project from one message type to
       another -->
   <Topic>
     <Name>example</Name>
     <MessageType>json</MessageType>
     <Key>/id</Key>
     <FileName>./sow/%n.sow</FileName>
   </Topic>
   <View>
     <!-- notice that nvfix topic named 'example'
          is *not* the same topic as the json
          topic named 'example' -->
     <Name>example</Name>
     <MessageType>nvfix</MessageType>
     <UnderlyingTopic>[json].[example]</UnderlyingTopic>
     <Projection>
        <Field>[json].[example]./id AS /id</Field>
     </Projection>
     <Grouping>
        <Field>[json].[example]./id</Field>
     </Grouping>
   </View>

   <!-- JOIN topics -->
   <Topic>
      <Name>ORDERS</Name>
      <MessageType>nvfix</MessageType>
      <Key>/OrderID</Key>
      <FileName>./sow/%n.sow</FileName>
   </Topic>
   <Topic>
      <Name>COMPANIES</Name>
      <MessageType>nvfix</MessageType>
      <Key>/CompanyId</Key>
      <FileName>./sow/%n.sow</FileName>
   </Topic>
   <View>
      <Name>TOTAL_COMPANY_VOLUME</Name>
      <UnderlyingTopic>
        <Join>[ORDERS]./Tick = [COMPANIES]./Tick</Join>
      </UnderlyingTopic>
      <FileName>./views/totalVolume.view</FileName>
      <MessageType>nvfix</MessageType>
      <Projection>
        <Field>[COMPANIES]./CompanyId</Field>
        <Field>[COMPANIES]./Tick</Field>
        <Field>[COMPANIES]./Name</Field>
        <Field>SUM([ORDERS]./Shares) AS /TotalVolume</Field>
      </Projection>
      <Grouping>
        <Field>[ORDERS]./Tick</Field>
      </Grouping>
    </View>
</SOW>
```

Example 9.3. View Example

9.4. SOW/ConflatedTopic

AMPS provides the ability to create ongoing snapshots of a SOW topic, called *conflated topics* (also called *topic replicas* in previous releases of AMPS). Topic replicas are updated on an interval, and store a snapshot of the current state of the world at each interval. This helps to manage bandwidth to clients that do not act on each update, such as a client UI that refreshes every second rather than with every update.

For compatibility with previous versions of AMPS, AMPS allows you to use `TopicReplica` as a synonym for `ConflatedTopic`.

Table 9.8. SOW/ConflatedTopic Parameters

Element	Description
`Name`	String used to define the name of the conflated topic. While AMPS doesn't enforce naming conventions, it can be convenient to name the conflated topic based on the underlying topic name. For example, if the underlying topic is `orders`, it can be convenient to name the conflated topic `orders-C`. If no `Name` is provided, AMPS accepts `Topic` as a synonym for `Name` to provide compatibility with versions of AMPS previous to 5.0.
`UnderlyingTopic`	String used to define the SOW topic which provides updates to the conflated topic. This must exactly match the name of a SOW topic.
`MessageType`	The message format of the underlying topic. This `MessageType` must be the `MessageType` of the provided `UnderlyingTopic`.
`Interval`	The frequency at which AMPS updates the data in the conflated topic. Default: `5 seconds`
`Filter`	Content filter that is applied to the underlying topic. Only messages that match the content filter are stored in the conflated topic.

```
<ConflatedTopic>
 <Topic>FastPublishTopic-C</Topic>
 <MessageType>nvfix</MessageType>
 <UnderlyingTopic>FastPublishTopic</UnderlyingTopic>
 <Interval>5s</Interval>
 <Filter>/region = 'A'</Filter>
</ConflatedTopic>
```

Chapter 10. Replication Destination

An AMPS replication target is defined within the `Replication` section of an AMPS configuration file. Within the `Replication` section, there are one or more `Destination` sections, each specifying a unique replication target. Table 10.1 contains a listing of the parameters for the `Destination` section in the `Replication` section of an AMPS configuration file.

Table 10.1. Replication Destination

Element	Description
`Destination`	Required parent tag, which defines a unique replication target.
`SyncType`	Defines how synchronization of `ack` messages is handled, either `sync` or `async`.
`Transport`	The message type and URI where messages will be replicated. Requires a `Type`, which must be “amps-replication” or “amps-replication-secure”. The element may have one or more `InetAddr` elements. AMPS supports multiple `Transport` items within a `Destination`. When multiple Transports are provided, AMPS interprets these as transports for redundant servers, listed in priority order. If AMPS cannot connect to any of the internet addresses in a transport, AMPS tries the next `Transport`, in the order in which the `Transport` items appear in the file. When AMPS has tried all of the `Transport` items, AMPS tries again at the beginning of the list of transports. To provide failover, use multiple `InetAddr` elements within a single `Transport` for servers that can use the same `Authenticator` context (that is, the same credentials provided with the same authentication scheme). Use multiple `Transport` elements if the failover servers require different authentication. **`Type`** The `Type` of a replication destination must always be `amps-replication` or `amps-replication-secure`. **`InetAddr`** A Transport for a replicaiton destination can contain one or more `InetAddr` elements. When a single `InetAddr` element is present, AMPS connects to that address for replication. When more than one `InetAddr` element is present, AMPS uses the list of addresses as a prioritized list of failover servers to provide high availability. The list is in priority order, with the most preferred server at the beginning of the list. Each time AMPS needs to make a connection for this `Destination`, AMPS starts with the first address in the list and tries each address in order until a connection succeeds. If no connection succeeds, AMPS waits for a timeout period and then either moves to the next `Transport` (if more than one `Transport` is present in the destination) or starts again with the first address in the list. Each time AMPS tries all of the addresses in the list without a successful connection, AMPS increases the timeout period between tries, up to a maximum timeout. The first time through the list, upon startup, AMPS gives addresses extra time, up to 60 seconds, to connect successfully.

Element	Description
	If no `InetAddr` is specified, then this `Destination` does not make an outgoing connection. Instead, this instance will wait for the remote instance specified in the `Destination` to connect, and replicate to that instance once the connection is established. **`Authenticator`** A `Transport` element within a `Destination` may contain an `Authenticator` element, which specifies a module that provides credentials to use when connecting to the destination. All of the `InetAddr` elements specified within a `Transport` use the same `Authenticator`. **`Certificate`** A `Transport` element that specifies `amps-replication-secure` as the transport type must provide a certificate to use for the SSL connection. **`PrivateKey`** A `Transport` element that specifies `amps-replication-secure` as the transport type must provide a private key file to use for the SSL connection.
`Group`	The group that the downstream destination is a member of. The `Group` of the downstream instance must match the `Group` specified in this destination, or AMPS reports an error and will not replicate to that destination. There is no default for this value. AMPS requires a `Group` for replication. If a `Group` is specified and no `Name` is specified, AMPS uses the value of the `Group` as the `Name`. Notice that the `Name` must be unique within an AMPS instance. If your replication configuration requires more than one `Destination` that replicates to the same `Group`, and does *not* want AMPS replication to treat all of those servers as identical, use the `Name` element instead of the `Group` element.
`Name`	The name of the destination. This name appears in the AMPS logs when AMPS logs a message about this destination. The `Name` must be unique in the AMPS instance. When not present, AMPS uses the `Group` provided as the destination `Name`. The `Name` should either the name or the `Group` of the remote instance. 60East recommends setting the `Name` only when your replication configuration replicates to more than one instance in a given group and the configuration does not need to treat the serers within the group as interchangeable. If it is important to replicate to a specific AMPS instance, rather than any server in the `Group`, set the `Name` rather than the `Group`, and use the `Name` of that instance. For example, if you have three servers in the `AMPS-LA` group, the server `AMPS-LA-1` would have separate `Destination` configurations for `AMPS-LA-2` and `AMPS-LA-3`. Those `Destination` configurations would use the `Name` of the remote server (`AMPS-LA-2` or `AMPS-LA-3`) rather than the `Group` that is common to all of the servers. There is no default for this value. If a `Group` is specified and no `Name` is specified, AMPS uses the value of the `Group` as the `Name` of the destination.
`Topic`	Defines the topic name to replicate. Requires a `Name` and `MessageType`. See the following table (Replication Destination : Topic Definition) for details.

Element	Description
`PassThrough`	Specifies source instances to pass through to this destination. The value of this element is a regular expression which is matched against the group name of the instance that sent the replication message to this instance. When the regular expression matches, the replication message is eligible for passthrough, and will be sent to the destination if the `Topic` specifications match the message.
`Compression`	Specifies whether to use compression for this destination. When set to `enabled`, AMPS compresses traffic to this destination. Default: `disabled`

A replication destination can contain any number of Topic definition elements. For simplicity in working with the configuration file, 60East recommends using a few `Topic` elements with regular expression patterns over large numbers of individual topic declarations.

Table 10.2. Replication Destination : Topic Definition

Element	Description
`Name`	The name of the topic to replicate. The `Name` can be either a literal topic name or a regular expression. When `Name` is a literal topic, a topic with that name and the specified message type must be captured in a transaction log. When `Name` is a regular expression, only topics that match the expression., match the message type, and are present in a transaction log are replicated.
`MessageType`	The message type of the topic to replicate.
`Filter`	A content filter to apply to the topics. When present, only messages that match the filter are replicated. This filter follows the standard AMPS filter syntax.
`IncludeValida-tion`	The set of configuration checks to validate for this topic. Default: All validation options listed below are included by default.
`ExcludeValida-tion`	The set of configuration checks to exclude for this topic. If the same check appears in both `IncludeValidation` and `ExcludeValidation`, `ExcludeValidation` takes precedence and the check will not be run. Default: None of the validation options listed below are excluded by default.

AMPS supports the following automatic configuration validation checks:

Table 10.3. Replication Configuration Validation

Check	Validates
`txlog`	The topic is contained in the transaction log of the remote instance.
`replicate`	The topic is replicated from the remote instance back to this instance.
`sow`	If the topic is a SOW topic in this instance, it must also be a SOW topic in the remote instance.
`cascade`	The remote instance must enforce the same set of validation checks for this topic as this instance does.

Check	Validates
`queue`	If the topic is a queue in this instance, it must also be a queue in the remote instance. This option cannot be excluded.
`keys`	If the topic is a SOW topic in this instance, it must also be a SOW topic in the remote instance and the SOW in the remote instance must use the same `Key` definitions.
`replicate_filter`	If this topic uses a replication filter, the remote instance must use the same replication filter for replication back to this instance.
`queue_underlying`	If the topic is a queue in this instance, it must use the same underlying topic definition and filters in the remote instance. This option cannot be excluded.

```
<Replication>
  <Destination>
    <Name>amps-2</Name>
    <Group>Data-Center-NYC-1</Group>
    <Topic>
      <Name>ORDER_STATE-Replication</Name>
      <MessageType>xml</MessageType>
    </Topic>
    <Topic>
      <Name>REFERENCE_INFO-.*</Name>
      <MessageType>json</MessageType>
      <Filter>/state = 'published'</Filter>
    </Topic>
    <SyncType>sync</SyncType>
    <Compression>enabled</Compression>
    <Transport>
      <Type>amps-replication</Type>
      <InetAddr>interface1.example.com:19005</InetAddr>
      <InetAddr>interface2.example.com:19080</InetAddr>
      <Authenticator>
         <Module>my-credentials-store-module</Module>
      </Authenticator>
    </Transport>
    <PassThrough>Data-Center-(ORD|HKG)-.*</PassThrough>
  </Destination>
  <Destination>
    <Name>NYC-View-Server</Name>
    <Group>Data-Center-NYC-1</Group>
    <Topic>
      <Name>ORDER_STATE</Name>
      <MessageType>json</MessageType>
      <ExcludeValidation>replicate,cascade,sow</ExcludeValidation>
    </Topic>
    <SyncType>async</SyncType>
    <Compression>enabled</Compression>
    <Transport>
      <Type>amps-replication</Type>
      <InetAddr>view-server-a.example.com:19005</InetAddr>
      <InetAddr>view-server-b.example.com:19080</InetAddr>
    </Transport>
  </Destination>
</Replication>
```

Example 10.1. Replication Example

Chapter 11. Transaction Log

AMPS includes the ability to record and replay messages. This capability can be used by applications for durable subscriptions, reliable publish, and historical replay. The AMPS transaction log is also the foundation of the high availability features in AMPS. To enable message recording and replay, configure a `TransactionLog` to keep a journal of messages published to an AMPS instance. The *Transactional Messaging and Bookmark Subscriptions* chapter in the AMPS User Guide covers how to use the transaction log for historical replay, durable publish, and durable subscriptions. The *Replication and High Availability* chapter in the *AMPS User Guide* covers the use cases where a `TransactionLog` can be used to maximize the up-time of your AMPS instance.

Table 11.1. TransactionLog Configuration Parameters

Element	Description
`JournalDirectory`	Filesystem location where journal files will be stored.
`JournalArchiveDirectory`	File system location where journal files are archived.
`PreallocatedJournalFiles`	The number of journal files AMPS will create as part of the server start-up. *Default: 2. Minimum: 1*
`JournalSize`	Sets the target size for AMPS to use when calculating the size of journal files. AMPS allocates journal files based on the size of an internal buffer. This option sets the target size for the journal file: AMPS will use the smallest file size that is an even multiple of the internal buffer *without going under* the specified `JournalSize`. Notice that AMPS does not grow journal files once they are allocated. When a journal file is full, AMPS uses the next journal file. AMPS accepts `MinJournalSize` as a synonym for `JournalSize`. *Default: 1GB. Minimum: 10M*
`Topic`	The topic to include in the transaction log. When no `Topic` is specified, AMPS initializes transaction log management for the instance, but does not persist messages.If a `Topic` is specified, then all messages which match exactly the specified topic or regular expression will be included in the transaction log. If you want all topics of a specific message type to be persisted, use the regular expression `.*` for the name of the topic. Multiple `Topic` elements can be included in a `TransactionLog` element.
`FlushInterval`	The interval at which messages will be flushed the journal file during periods of slow activity. *Default: 100ms Maximum: 100ms Minimum: 30us*
`MetadataIndexing`	Specifies whether to create journal index files for the journal. When set to `persistent`, AMPS creates journal index files. When set to `transient`, AMPS does not create journal index files. *Default: persistent*
`O_DIRECT`	Where supported, `O_DIRECT` will perform DMA directly from/to physical memory to a userspace buffer. Having this enabled can improve AMPS performance, however not all devices support `O_DIRECT`. *Default: enabled.*

Element	Description
DEPRECATED `BatchSize`	*This element is no longer necessary in releases of AMPS 4.0 and greater. If this element is present in the configuration, AMPS emits a deprecation warning and ignores the configured value.*

Example 11.1 demonstrates a transaction log where the journal file will be written to `./amps/journal`. When AMPS starts, a single journal file will be pre-allocated as noted by the `PreallocatedJournalFiles` setting; and when the first journal file is completely full, 128 new journal files will be created. This journal is going to contain only those messages which match the topic `orders` and also have a message type of `fix`. If, at any time, there is 40us of inactivity while there is data to be flushed to the journal file, AMPS will proactively flush the data to the file.

```
<AMPSConfig>
...

  <TransactionLog>
    <JournalDirectory>./amps/journal/</JournalDirectory>
    <PreallocatedJournalFiles>1</PreallocatedJournalFiles>
    <MinJournalSize>10MB</MinJournalSize>
    <Topic>
      <Name>orders</Name>
      <MessageType>nvfix</MessageType>
      <Filter>/price > 5</Filter>
    </Topic>
    <Topic>
      <Name>LOGGED_.*</Name>
      <MessageType>json</MessageType>
    </Topic>
    <FlushInterval>40ms</FlushInterval>
  </TransactionLog>

...
</AMPSConfig>
```

Example 11.1. Transaction Log Configuration Example

Chapter 12. Authentication

The `Authentication` element specifies the module to use for validating user identity. AMPS allows you to set the default `Authentication` for the instance as a whole, and also to set the `Authentication` on each `Transport` individually.

`Authentication` elements are not required. The instance authentication defaults to using the `amps-default-authentication-module` if no `Authentication` element is specified for the instance. An individual `Transport` defaults to using the instance `Authentication` if no `Authentication` element is provided for that `Transport`.

Table 12.1. Authentication Parameters

Name	*Description*
`Module`	The element specifies the name of the module that will be used for authentication. The value of this element must be the name of an authentication module loaded in the Modules section of the configuration file or one of the authentication modules that AMPS loads by default. By default, AMPS loads the authentication modules listed in Table 12.2.
`Options`	A list of supported features for the implemented library. AMPS allows you to pass options to the module by specifying elements within the `Options` element. The exact options that the module requires, if any, are determined by the creator of the module.

AMPS loads the following authentication modules by default:

Table 12.2. AMPS default authentication modules

Module Name	Policy
`amps-default-authentication-module`	Authenticate any user, regardless of the credentials provided. Does not provide the user name to AMPS by default, and does not allow implicit authentication by default. The `amps-default-authentication-module` accepts two options: • `AllowSpoofing`. When set to `enabled`, this module provides the user name to AMPS. The `AllowSpoofing` option is set to `disabled` by default. • `RequireLogon`. When set to enabled, this module does not allow implicit logon. Connections must explicitly logon or the module will refuse to authentication them. This option is set to `enabled` by default.
`amps-implicit-authentication-module`	Authenticate any user, regardless of the credentials provided. Allows implicit authentication. Does not provide the user name to AMPS by default. This module accepts the following option:

Module Name	Policy
	• `AllowSpoofing`. When set to `enabled`, this module provides the user name to AMPS. The `AllowSpoofing` option is set to `disabled` by default. This module is provided to mimic the default behavior of the `amps-default-authentication-module` in versions prior to 5.0. To restore that behavior, set `amps-implicit-authentication-module` to the `Authenticator` for the instance.
`amps-default-no-authentication-module`	Do not authenticate any user.

Chapter 13. Entitlement

The `Entitlement` element specifies the module to use for validating permissions to resources within AMPS. AMPS allows you to set the default `Entitlement` for the instance as a whole, and also to set the `Entitlement` on each Transport individually.

`Entitlement` elements are not required. The instance authentication defaults to using the `amps-default-entitlement-module` if no `Entitlement` element is specified for the instance. An individual `Transport` defaults to using the instance `Entitlement` if no `Entitlement` element is provided for that `Transport`.

Table 13.1. Entitlement Parameters

Name	*Description*
`Module`	The element specifies the name of the module that will be used for entitlement. The value of this element must be the name of an entitlement module loaded in the Modules section of the configuration file or one of the entitlement modules that AMPS loads by default. By default, AMPS loads the entitlement modules listed in Table 13.2.
`Options`	A list of options to provide to the module for this instance of the module. AMPS allows you to pass options to the module by specifying elements within the `Options` element. The exact options that the module requires, if any, are determined by the creator of the module.

AMPS loads the following entitlement modules by default:

Table 13.2. AMPS default entitlement modules

Module Name	Policy
`amps-default-entitlement-module`	Allow all permissions to every user.
`amps-default-no-entitlement-module`	Deny all permissions to every user.

Chapter 14. Actions

AMPS includes the ability to perform administrative tasks in response to Linux signals or on a set schedule. The `Actions` element allows you to specify these actions and when they occur.

The `Actions` element contains one or more `Action` elements. An `Action` element contains an `On` element, which tells AMPS when to perform the task, and a `Do` element, which tells AMPS what task to perform.

14.1. Setting when an Action Runs

This section describes the options for configuring when AMPS runs a given action.

Running an Action on a Schedule

AMPS provides the `amps-action-on-schedule` module for running actions on a specified schedule.

The options provided to the module define the schedule on which AMPS will run the actions in the Do element.

Table 14.1. Parameters for Scheduling Actions

Parameter	Description
`Every`	Specifies a recurring action that runs whenever the time matches the provided specification. Specifications can take three forms: • *Timer action*. A specification that is simply a duration, such as `4h` or `1d`, creates a timer action. AMPS starts the timer when the instance starts. When the timer expires, AMPS runs the action and resets the timer. • *Daily action*. A specification that is a time of day, such as `00:30` or `17:45`, creates a daily action. AMPS runs the action every day at the specified time. AMPS uses a 24 hour notation for daily actions. • *Weekly action*. A specification that includes a day of the week and a time, such as `Saturday at 11:00` or `Wednesday at 03:30` creates a weekly action. AMPS runs the action each week on the day specified, at the time specified. AMPS uses a 24 hour notation for weekly actions. AMPS accepts both local time and UTC for time specifications. To use UTC, append a `Z` to the time specifier. For example, the time specification `11:30` is 11:30 AM local time. The time specification `11:30Z` is 11:30 AM UTC.
`Name`	The name of the schedule. This name appears in log messages related to this schedule. Default: `unknown`

This module does not add any variables to the AMPS context.

Running an Action in Response to a Signal

AMPS provides the `amps-action-on-signal` module for running actions when AMPS receives a specified signal.

The module requires the `Signal` parameter:

Table 14.2. Parameters for Responding to Signals

Parameter	Description
`Signal`	Specifies the signal to respond to. This module supports the standard Linux signals. Configuring an action uses the standard name of the signal. For example, to configure an action to `SIGUSR1`, the value for the `Signal` element is `SIGUSR1`. To configure an action for `SIGHUP`, the value for the `Signal` element is `SIGHUP` and so on. AMPS reserves `SIGQUIT` for producing minidumps, and does not allow this module to override `SIGQUIT`. AMPS registers actions for several signals by default. See the section called "Default Signal Actions" for details.

This module does not add any variables to the AMPS context.

Actions can be used to override the default signal behavior for AMPS.

Default Signal Actions

By default, AMPS registers the following actions for signals.

Table 14.3. Default Actions

On Event	Action
`SIGUSR1`	`amps-action-do-disable-authentication`
`SIGUSR1`	`amps-action-do-disable-entititlement`
`SIGUSR2`	`amps-action-do-enable-authentication`
`SIGUSR2`	`amps-action-do-enable-entitlement`
`SIGINT`	`amps-action-do-shutdown`
`SIGTERM`	`amps-action-do-shutdown`
`SIGHUP`	`amps-action-do-shutdown`

The actions in the table above can be be overridden by creating an explicit action in the configuration file.

AMPS reserves `SIGQUIT` to perform the action `amps-action-do-minidump`. This behavior is reserved, and cannot be overridden.

Running an Action on Startup or Shutdown

AMPS includes modules to run actions when AMPS starts up or shuts down.

The `amps-action-on-startup` module runs actions as the last step in the startup sequence. The `amps-action-on-shutdown` module runs actions as the first step in the AMPS shutdown sequence.

In both cases, actions run in the order that the actions appear in the configuration file.

These modules do not require any parameters.

These modules do not add any variables to the AMPS context.

Runnning an Action on Client Logon

AMPS provides the `amps-action-on-logon` module for running actions when a user logs into an AMPS client.

This module does not require any parameters.

This module adds the following variables to the AMPS context:

Table 14.4. Context Variables for On Client Logon

Variable	Description
`AMPS_CLIENT_NAME`	The name of the AMPS Client.
`AMPS_CONNECTION_NAME`	The name of the AMPS connection.

Running an Action on Client Connection

AMPS provides modules for running actions on the connection or disconnection of an AMPS client.

The `amps-action-on-disconnect-client` runs actions once an AMPS client instance disconnects. The `amps-action-on-connect-client` runs actions once an instance of an AMPS client successfully connects.

These modules do not require any parameters.

These modules add the following variables to the AMPS context.

Table 14.5. Context Variables for On Connect and Disconnect Client

Variable	Description
`AMPS_CLIENT_NAME`	The name of the AMPS client.
`AMPS_CONNECTION_NAME`	The name of the AMPS connection.

Running an Action on Message Delivery

AMPS provides modules to run actions when AMPS delivers a message to subscribers. The basic flow of AMPS messaging is to first receive a published message, find the subscriber(s) to which this message will be sent, then deliver the message.

The `amps-action-on-deliver-message` runs actions when AMPS delivers a message to subscribers.

This module requires the `MessageType` and the `Topic` of the message that has been delivered:

Table 14.6. Parameters for On Deliver Message

Parameter	Description
`MessageType`	The message type of the topic to monitor for message delivery. There is no default for this parameter.
`Topic`	The name of the topic to monitor for message delivery. This parameter supports regular expressions. There is no default for this parameter.

This module adds the following variables to the AMPS context:

Table 14.7. Context Variables for On Deliver Message

Variable	Description
`AMPS_TOPIC`	The topic of the message.
`AMPS_DATA`	The data the message contains.
`AMPS_DATA_LENGTH`	The length of the data the message contains.
`AMPS_BOOKMARK`	The bookmark associated with this message. This is an empty string if the message does not have a bookmark.
`AMPS_CLIENT_NAME`	The name of the client to which this message was delivered.

Running an Action on Message Publish

AMPS provides modules to run actions when a message is published to AMPS. The basic flow of AMPS messaging is to first receive a published message, find the subscriber(s) to which this message will be sent, then deliver that message to the subscriber(s).

The `amps-action-on-publish-message` runs actions as soon as a message is published to AMPS.

This module requires the `MessageType` and the `Topic` of the message that was published. In addition to that, this module also accepts an optional `MessageSource` parameter:

Table 14.8. Parameters for On Publish Message

`MessageType`	The message type of the topic to monitor for publishes. There is no default for this parameter.
`Topic`	The name of the topic to monitor for publishes. This parameter supports regular expressions. There is no default for this parameter.
`MessageSource`	The source to monitor for publishes. The source of the message defaults to `all`, which monitors both publishes directly to this AMPS instance and messages received via replication. This parameter also accepts `local` for when the message source is published directly to this AMPS instance and `replicated` for messages received via replication.
`Filter`	Sets the filter to apply. Only messages that match this filter will cause the action to run.

This module adds the following variables to the AMPS context:

Table 14.9. Context Variables for On Publish Message

Variable	Description
`AMPS_TOPIC`	The topic of the message.
`AMPS_DATA`	The data the message contains.
`AMPS_DATA_LENGTH`	The length of the data that the message contains.
`AMPS_BOOKMARK`	The bookmark associated with this message.
`AMPS_TIMESTAMP`	The time at which the message was processed by AMPS.
`AMPS_CLIENT_NAME`	The name of the client from which the message was published.

Running an Action on OOF Message

When a record that previously matched a subscription has been updated so that the record no longer matches its subscription, AMPS sends an out-of-focus (OOF) message to let subscribers know that their record no longer matches the subscription. With `amps-action-on-oof-message`, you can enter a subscription within AMPS and run actions when an OOF message for that subscription is produced.

This module requires the following parameters:

Table 14.10. Parameters for On OOF Message

Parameter	Description
`MessageType`	The message type of the topic to monitor for OOF messages. This parameter supports regular expressions. There is no default for this parameter.
`Topic`	The topic to monitor for OOF messages.The topic specified must be a SOW topic, view, or conflated topic. This parameter supports regular expressions. There is no default for this parameter.
`Filter`	Set the filter to apply. This filter forms the internal subscription for which OOF messages will be generated.
`Type`	The type of OOF message to take action on. (see Table 14.11) Defaults to `all`.

Table 14.11. OOF message types for amps-action-on-oof-message

Parameter	Description
`match`	Take action on OOF messages generated because message no longer matches filter.
`delete`	Take action on OOF messages generated because message has been removed from the SOW.
`expire`	Take action on OOF messages generated because the message expired from the SOW.
`all`	Take action on all of the above types.

This module adds the following variables to the AMPS context:

Table 14.12. Context Variables for On OOF Message

Variable	Description
`AMPS_TOPIC`	The topic of the OOF message.
`AMPS_DATA`	The data of the OOF message.
`AMPS_DATA_LENGTH`	The length of the data of the OOF message.
`AMPS_PREVIOUS_DATA`	The data previously contained from the updated record.
`AMPS_PREVIOUS_DATA_LENGTH`	The length of the data previously contained from the updated record.

Running an Action on Minidump

AMPS provides the `amps-action-on-minidump` module for running actions when AMPS generates a minidump.

This module does not require parameters.

This module adds the following variable to the AMPS context:

Table 14.13. Context Variable for On Minidump

Variable	Description
`AMPS_MINIDUMP_PATH`	The path to where the minidump is created.

Running an Action on Offline Start or Stop

AMPS provides modules to run actions when an AMPS client is marked as a slow client, and also for when the AMPS client catches up to no longer be subject to slow client offlining.

Slow client offlining is a feature in AMPS that reduces the memory resources consumed by slow clients. More on this feature can be found in ???.

The `amps-action-on-offline-start` module runs actions as the first step when AMPS's result set reaches its disk limit and has to disconnect the client. The `amps-action-on-offline-stop` module runs actions as AMPS is no longer subject to slow client offlining.

In both cases, actions run in the order that the actions appear in the configuration file.

Both modules do not require any parameters.

Both modules add the following variables to the AMPS context:

Table 14.14. Context Variables for On Offline Start and Stop

Variable	Description
`AMPS_CLIENT_NAME`	The name of the AMPS client.
`AMPS_CONNECTION_NAME`	The name of the AMPS connection.

Running on Action on SOW Message Deletion

AMPS provides a module to run an action when a message is deleted from a topic in the SOW.

The amps-action-on-sow-delete-message module monitors a topic for deletions from the SOW. The action runs once for each message that is deleted in the matching topic.

Table 14.15. Parameters for On SOW Message Deletion

`MessageType`	The message type of the topic to monitor for messages. There is no default for this parameter.
`Topic`	The name of the topic to monitor for messages. This parameter *does not* support regular expressions. The topic name must be one of the topics in the SOW (either a topic in the SOW, a view, a conflated topic, or a queue). There is no default for this parameter.

The module adds the following variables to the AMPS context:

Table 14.16. Context Variables for On SOW Message Delete

Variable	Description
`AMPS_TOPIC`	The topic of the message that expired the alert.
`AMPS_DATA`	The current data of the message.
`AMPS_DATA_LENGTH`	The length of the current data of the message, in bytes.

Running an Action on SOW Message Expiration

AMPS provides a module to run an action when a message expires from a topic in the SOW.

The `amps-action-on-sow-expire-message` module monitors a topic for expirations. The action runs once for each message that expires in the matching topic. Notice, in particular, that this includes monitoring messages that expire from the queue, which are presented as SOW expirations to this module.

Table 14.17. Parameters for On SOW Message Expiration

`MessageType`	The message type of the topic to monitor for messages. There is no default for this parameter.
`Topic`	The name of the topic to monitor for messages. This parameter *does not* support regular expressions. The topic name must be one of the topics in the SOW (either a topic in the SOW, a view, a conflated topic, or a queue). There is no default for this parameter.

The module adds the following variables to the AMPS context:

Table 14.18. Context Variables for On SOW Message Expire

Variable	Description
`AMPS_TOPIC`	The topic of the message that expired the alert.
`AMPS_DATA`	The current data of the message.
`AMPS_DATA_LENGTH`	The length of the current data of the message, in bytes.

Running an Action on Message Condition Timeout

AMPS provides a module to run an action when a message in a SOW topic meets a specific condition for longer than a specified period of time. For example, an action might be configured to publish a message to an `Alerts` topic if an order is unprocessed for more than a specified timeout.

The `amps-action-on-message-condition-timeout` monitors a SOW topic for messages that match a filter and triggers an action for each message that remains matched on that filter for at least the specified duration.

This module uses the Out-of-Focus notification (OOF) mechanism. When a message matches the specified topic and filter, the module begins tracking that message. If no OOF notification is received for that message within the specified timeout, the action runs for that message.

The module tracks each message that matches the filter individually, and will run once for each message that exceeds the timeout.

While the AMPS server is running, this action will trigger exactly once for each message after it reaches the timeout period. When AMPS restarts, if a message that had previously triggered this action still exists in the SOW topic (and still matches the filter provided, if any), the action will run for that message immediately after module initializes on restart.

Table 14.19. Parameters for On Message Condition Timeout

`MessageType`	The message type of the topic to monitor for messages. There is no default for this parameter.
`Topic`	The name of the topic to monitor for messages. This parameter *does not* support regular expressions. The topic name must be one of the topics in the SOW (either a topic in the SOW, a view, or a conflated topic). Queues are not supported. There is no default for this parameter.
`Duration`	The amount of time to wait for an OOF notification for the message before running the action.
`Filter`	Sets the filter to apply. Only messages that match this filter will be monitored by this action. If no filter is provided, every message of the specified message type in topics that match the `Topic` value will be monitored.

The module adds the following variables to the AMPS context:

Table 14.20. Context Variables for On Message Condition Timeout

Variable	Description
`AMPS_TOPIC`	The topic of the message that triggered the alert.
`AMPS_DATA`	The current data of the message.
`AMPS_DATA_LENGTH`	The length of the current data of the message, in bytes.
`AMPS_BOOKMARK`	The bookmark of the message. Empty if ther eis no bookmark for the message.
`AMPS_TIMESTAMP`	The timestamp at which the module began tracking the message.
`AMPS_CLIENT_NAME`	The client name of the current value of the message.

Variable	Description
`AMPS_SOW_KEY`	The current SowKey for the message.

14.2. Defining the Action to Take

This section describes the default modules for specifying what AMPS does when an action runs.

Rotate Log Files

AMPS provides the following module for rotating log files. AMPS loads this module by default:

Table 14.21. Managing Logs

Module Name	Does
`amps-action-do-rotate-logs`	Rotates logs that are older than a specified age, for log types that support log rotation. Rotating a log involves closing the log and opening the next log in sequence. AMPS will use the name specifier provided in the AMPS configuration for the new log file. This may overwrite the current log file if the specifier results in the same name as the current log file.

This module does not require options.

This module does not add any variables to the AMPS context:

Manage the Statistics Database

AMPS provides the following modules for managing the statistics database. As a maintenance strategy, 60East recommends truncating statistics on a regular basis. This frees space in the database file, which will be reused as new statistics are generated. It is generally not necessary to vacuum statistics unless you have changed your retention policy so that less data is retained between truncation operations. With regular truncation, the statistics database file will usually stabilize at the correct size to hold the amount of data your application generates between truncation operations.

AMPS loads these modules by default.

Table 14.22. Managing Logs

Module Name	Does
`amps-action-do-truncate-statistics`	Removes statistics that are older than a specified age. This frees space in the statistics file, but does not reduce the size of the file.
`amps-action-do-vacuum-statistics`	Remove unused space in the statistics file to reduce the size of the file. In general, it is not necessary to remove unused space in the statistics file. This operation can be expensive, and query access to the statistics database can be un-

Module Name	Does
	available for an extended period of time if the file is large. If storage space is in high demand, and the interval at which the file is vacuumed has been reduced, removing space from the file can sometimes reduce the space needs. 60East recommends using this action only in long-running AMPS environments where space is at a premium, and scheduling the action during times when it is acceptable for monitoring of the system to be unavailable while the file is processed.

The `amps-action-do-truncate-statistics` module requires an `Age` parameter that specifies the age of the statistics to process.

Table 14.23. Parameters for Managing Statistics

Parameter	Description
`Age`	Specifies the age of the statistics to remove. The module processes any file older than the specified `Age`. For example, when the `Age` is `5d`, the module removes statistics that are older than 5d. There is no default for this parameter.

These modules do not add any variables to the AMPS context.

Manage Journal Files

AMPS provides the following modules for managing journal files. AMPS loads these modules by default:

Table 14.24. Managing Journals

Module Name	Does
`amps-action-do-archive-journal`	Archives journal files that are older than a specified age to the `JournalArchiveDirectory` specified for the transaction log.
`amps-action-do-compress-journal`	Compresses journal files that are older than a specified age.
`amps-action-do-remove-journal`	Deletes journal files that are older than a specified age.

Each of these modules requires an `Age` parameter that specifies the age of the journal files to process.

AMPS will only remove journal files that are no longer needed by the instance. AMPS ensures that all replays from a journal file are complete, all queue messages in that journal file have been delivered (and acknowledged, if required), and all messages from a journal file have been successfully replicated before removing the file.

Table 14.25. Parameters for Managing Journals

Parameter	Description
`Age`	Specifies the age of files to process. The module processes any file older than the specified `Age`. For example, when the `Age` is `5d`, only files that have not been written to for longer than 5 days will be processed by the module. AMPS does not

Parameter	Description
	process the current log file, or files that are being used for replay, files that are being used for replication, or files that contain unacknowledged and unexpired messages in a queue; even if the file has been inactive for longer than the `Age` parameter. AMPS does not allow gaps in the journal files, so it will only remove a given file if all previous files have been removed. There is no default for this parameter.

These modules do not add any variables to the AMPS context.

Removing Files

AMPS provides the following module for removing files. Use this action to remove error log files that are no longer needed. AMPS loads this module by default. This action cannot be used to safely remove journal files (also known as transaction log files). For those files, use the journal management actions described in the section called “Manage Journal Files”.

This action removes files that match an arbitrary pattern. If the pattern is not specified carefully, this action can remove files that contain important data, are required for AMPS, or are required by the operating system.

This action cannot be used to safely remove journal files. Use the actions in the section called “Manage Journal Files” to manage journal files.

Table 14.26. Removing Files

Module Name	Does
`amps-action-do-remove-files`	Removes files that match the specified pattern that are older than the specified age. This action accepts an arbitrary pattern, and removes files that match that pattern. While AMPS attempts to protect against deleting journal files, using a pattern that removes files that are critical for AMPS, for the application, or for the operating system may result in loss of data. The module does not recurse into directories. It skips open files. The module does not remove AMPS journals (that is, files that end with a `.journal` extension), and reports an error if a file with that extension matches the specified `Pattern`. The commands to remove files are executed with the current permissions of the AMPS process.

This module requires an `Age` parameter that specifies the age of the files to remove, as determined by the update to the file. This module also requires a `Pattern` parameter that specifies a pattern for locating files to remove.

Table 14.27. Parameters for Removing Files

Parameter	Description
`Age`	Specifies the age of files to process. The module removes any file older than the specified `Age` that matches the specified `Pattern`. For example, when the `Age` is `5d`, only files that have not modified within 5 days and that match the pattern will be processed by the module.

Parameter	Description
	There is no default for this parameter.
`Pattern`	Specifies the pattern for files to remove. The module removes any files that match the specified `Pattern` that have not been modified more recently than the specified `Age`. This parameter is interpreted as a Unix shell globbing pattern. It is *not* interpreted as a regular expression. As with other parameters that use the file system, when the pattern specified is a relative path the parameter is interpreted relative to the current working directory of the AMPS process. When the pattern specified is an absolute path, AMPS uses the absolute path. There is no default for this parameter.
`Keep`	Specifies the number of files that meet the Age and Pattern criteria to retain. When this parameter is specified, AMPS will remove files matching the criteria, starting with the oldest files, and stop when the number of remaining files is the number specified in this parameter. There is no default for this parameter. When both `Keep` and `Count` are specified, AMPS will not remove any files if the number of files meeting the criteria is less than the number specified in the `Keep` parameter.
`Count`	Specifies the maximum number of files that meet the `Age` and `Pattern` criteria to remove. AMPS will remove files matching the criteria, starting with the oldest files, and stop when the number of files specified in this parameter have been removed. There is no default for this parameter. When both `Keep` and `Count` are specified, AMPS will not remove any files if the number of files meeting the criteria is less than the number specified in the `Keep` parameter.

This module does not add any variables to the AMPS context.

Deleting Messages from SOW

AMPS also provides modules for deleting SOW contents. The `amps-action-do-delete-sow` module deletes messages from the specified SOW topic.

This module requires the `MessageType`, `Topic`, and `Filter` parameters in order to delete the desired message.

Table 14.28. Parameters for Deleting SOW Messages

Parameter	Description
`MessageType`	The MessageType of the SOW topic or topics to delete from. There is no default for this parameter.
`Topic`	The name of the SOW topic from which to delete messages. This parameter supports regular expressions. There is no default for this parameter.
`Filter`	Set the filter to apply. Only messages matching that filter will be deleted.

This module does not add any variables to the AMPS context.

Compacting a SOW File

AMPS also provides provides a module for reducing the unused space in a SOW file. The `amps-action-do-compact-sow` module rearranges the messages in the SOW into a smaller amount of space, where possible.

This module can compact a specific SOW file, or the SOW files for every topic in the instance. When a `MessageType` and `Topic` are provided, this module compacts the SOW file for that topic. Otherwise, the module compacts the file for all topics in the SOW.

While messages are being added or updated within a topic in the SOW, AMPS reuses free space as possible: it is not necessary to compact the SOW file during most normal operation. This action is most useful after an activity peak that leaves a large amount of unneeded space in the file, or in installations where space is at a premium. Depending on the file size, the number of topics to be compacted, and the amount of free space, the reogranization that this operation performs may require a noticeable amount of I/O bandwidth. 60East recommends that this action run during a maintenance window or in response to a critical lack of disk space.

Table 14.29. Parameters for Deleting SOW Messages

Parameter	Description
`MessageType`	The MessageType of the SOW topic or topics to delete from. This option must be specified if the `Topic` is provided. There is no default for this parameter.
`Topic`	The name of the SOW topic from which to delete messages. This option must be specified if the `MessageType` is provided. There is no default for this parameter.

This module does not add any variables to the AMPS context.

Querying a SOW Topic

AMPS provides a module for querying a SOW topic. The `amps-action-do-query-sow` queries the SOW topic, and stores the first message returned by the SOW query into a user-defined variable.

This module requires the `MessageType`, `Topic`, and `Filter` parameters to identify the query to run. This module requires the `CaptureData` parameter in order to be able to store the result of the query.

Table 14.30. Parameters for Querying SOW Messages

Parameter	Description
`MessageType`	The message type of the topic to query. There is no default for this parameter
`Topic`	The name of the topic to query. This topic must be a SOW topic, a view, a queue, or a conflated topic. There is no default for this parameter. This parameter supports regular expressions.
`Filter`	Set the filter to apply. If a `Filter` is present, only messages matching that filter will be returned by the query.

Parameter	Description
`CaptureData`	Sets the name of the variable within which AMPS will store the first message returned.
`DefaultData`	If no records are found, AMPS stores the `DefaultData` in the variable specified by `CaptureData`.
`OrderBy`	An `OrderBy` expression to use to order the results returned by the query. For example, to order in descending order of the `/date` field in the messages, you would provide an `OrderBy` option of `/date DESC`.

Once you query messages from the SOW topic, you can use the captured data in other actions. The example below uses `amps-action-do-query-sow` to query the SOW on a schedule in order to echo messages to the log for diagnostic purposes:

```
<Actions>
  <Action>
    <On>
      <Module>amps-action-on-schedule</Module>
      <Options>
        <Every>Saturday at 23:59</Every>
        <Name>Diagnostic_Schedule</Name>
      </Options>
    </On>
    <Do>
      <Module>amps-action-do-query-sow</Module>
      <Options>
        <MessageType>xml</MessageType>
        <Topic>SOW_TOPIC</Topic>
        <Filter>/Trans/Order/@Oname = 'PURCHASE'</Filter>
        <CaptureData>AMPS_DATA</CaptureData>
      </Options>
    </Do>
    <Do>
      <Module>amps-action-do-extract-values</Module>
      <Options>
        <MessageType>xml</MessageType>
        <Data>{{AMPS_DATA}}</Data>
        <Value>SAVED_VARIABLE=/Value</Value>
      </Options>
    <Do>
      <Module>amps-action-do-echo-message</Module>
      <Options>
        <Message>{{SAVED_VARIABLE}} was in the message</Message>
      </Options>
    </Do>
  </Action>
</Actions>
```

Manage Security

AMPS provides modules for managing the security features of an instance.

Authentication and entitlement can be enabled or disabled, which is useful for debugging or auditing purposes. You can also reset security and authentication, which clears the AMPS internal caches and gives security and authentication modules the opportunity to reinitialize themselves, for example, by re-parsing an entitlements file.

AMPS loads the following modules by default:

Table 14.31. Security Modules

Module Name	Does
`amps-action-do-disable-authentication`	Disables authentication for the instance.
`amps-action-do-disable-entitlement`	Disables entitlement for the instance.
`amps-action-do-enable-authentication`	Enables authentication for the instance.
`amps-action-do-enable-entitlement`	Enables entitlement for the instance.
`amps-action-do-reset-authentication`	Resets authentication by clearing AMPS caches and reinitializing authentication
`amps-action-do-reset-entitlement`	Resets entitlement by clearing AMPS caches and reinitializing entitlement

These modules require no parameters. The `amps-action-do-reset-authentication` module and the `amps-action-do-reset-entitlement` module accept an optional `Transport` parameter which specifies the transport to reset.

Table 14.32. Parameters for Reset Authentication or Entitlement

Parameter	Description
`Transport`	The `Name` of the transport for which to reset authentication or entitlements. If no `Name` is provided, these modules affect all transports.

These modules do not add any variables to the AMPS context.

Enable and Disable a Transport

AMPS provides modules that can enable and disable specific transports. The `amps-action-do-enable-transport` module enables a transport. The `amps-action-do-disable-transport` module disables a transport.

Table 14.33. Transport Action Modules

Module Name	Does
`amps-action-do-enable-transport`	Enables a specific transport.
`amps-action-do-disable-transport`	Disables a specific transport.

Both modules require the name of the transport to disable or enable.

Table 14.34. Parameters for Managing Transports

Parameter	Description
`Transport`	The `Name` of the transport to enable or disable.

Parameter	Description
	If no `Name` is provided, the module affects all transports.

Both modules do not add any variables to the AMPS context.

Publishing Messages

The `amps-action-do-publish-message` module publishes a message into a specified topic.

Publishes from this action are treated as publishes from an AMPS client inside the AMPS engine. This means that:

- There are no user credentials associated with the publish, so entitlements are not applied.
- There is no special handling for the publish. The publish is recorded in the transaction log exactly as if it arrived from outside of the instance, and is processed within the instance as if the had arrived from an external publisher.

This action is treated by the AMPS engine as a publish from an internal AMPS client.When an `amps-action-do-publish-message` runs in response to the `amps-action-on-publish-message` event or the `amps-action-on-deliver-message` event, use caution when the message published from this action could cause the event to trigger again.

This warning includes cases where the action publishes to a topic directly monitored by the action, cases where the action monitors a view and publishes to an underlying topic of the view. The warning also applies to configurations in which two or more actions "cross publish" to topics that are monitored by the other action. An example of the last case is an action that monitors `TopicOne` and publishes to `TopicTwo`, while another action monitors `TopicTwo` and publishes to `TopicOne`.

The result of a configuration like the ones described above is called a *publish loop*. AMPS does not support unterminated publish loops or loops that produce a large number of cycles before terminating.

To publish a message, this module requires the `MessageType`, a `Topic` to publish on, and also the `Data` that the message will contain.

Table 14.35. Parameters for Publishing Messages

Parameter	Description
`MessageType`	The MessageType for the topic. There is no default for this parameter.
`Topic`	The topic of the message being published.
`Data`	The data that the message will contain.
`Delta`	Whether to use a delta publish. When this option is present, and the value is `true`, the action will use a delta publish. When no value is specified, this option is `false`.
`UpdateOnly`	Specifies whether a delta publish is allowed to insert a record, or only update a record. When a delta publish is specified (that is, `Delta` is `true`), and this option is set to `true`, AMPS will only accept the publish if there is a record present to be updated. When no value is specified, this option is `false`.

This action is treated by the AMPS engine as a publish from an internal AMPS client.When an `amps-action-do-publish-message` runs in response to the `amps-action-on-publish-message` event or the `amps-action-on-deliver-message` event, use caution when the message published from this action could cause the event to trigger again.

This includes both cases where the action publishes to a topic directly monitored by the action, cases where the action monitors a view and publishes to an underlying topic of the view, and cases where two or more actions each publish to a topic that is monitored by another action.

In effect, a configuration like the one described above creates a recursive call to the action: that recursion must terminate, and must terminate at a relatively low depth. (The exact limits depend on system capacity, message size, and so on).

This module does not add any variables to the AMPS context.

Manage Replication

AMPS provides modules for downgrading replication destinations that fall behind and upgrading them again when they catch up.

Table 14.36. Replication Modules

Module Name	Does
`amps-action-do-downgrade-replication`	Downgrades replication connections from synchronous to asynchronous if the age of the last acknowledged message is older than a specified time period.
`amps-action-do-upgrade-replication`	Upgrades previously-downgraded replication connections from asynchronous to synchronous if the age of the last acknowledged message is more recent than a specified time period. This action has no effect on replication destinations that are specified as `async` in the configuration file.

The modules determine when to downgrade and upgrade based on the age of the oldest message that a destination has not yet acknowledged. When using these modules, it is important that the thresholds for the modules are not set too close together. Otherwise, AMPS may repeatedly upgrade and downgrade the connection when the destination is consistently acknowledging messages at a rate close to the threshold values. To avoid this, 60East recommends that the `Age` set for the upgrade module is 1/2 of the age used for the downgrade module.

The `amps-action-do-downgrade-replication` module accepts the following options:

Table 14.37. Parameters for Downgrading Replication

Parameter	Description
`Age`	Specifies the maximum message age at which AMPS downgrades a replication destination to `async`. When this action runs, AMPS downgrades any destination for which the oldest unacknowledge message is older than the specified `Age`. For example, when the `Age` is `5m`, AMPS will downgrade any destination where a message older than 5 minutes has not been acknowledged. There is no default for this parameter.

Parameter	Description
`GracePeriod`	The approximate time to wait after start up before beginning to check whether to downgrade links. The `GracePeriod` allows time for other AMPS instances to start up, and for connections to be established between AMPS instances.

The `amps-action-do-upgrade-replication` module only applies to destinations configured as `sync` that have been previously downgraded. The module accepts the following options:

Table 14.38. Parameters for Upgrading Replication

Parameter	Description
`Age`	Specifies the maximum message age at which a previously-downgraded destination will be upgraded to `sync` mode. When this action runs, AMPS upgrades any destination that has been previously downgraded where the oldest unacknowledged message to AMPS is more recent than time value specified in the `Age` parameter. For example, if a destination has been downgraded to `async` mode and the `Age` is `2m`, AMPS will upgrade the destination when the oldest unacknowledged message to that destination is less than 2 minutes old. There is no default for this parameter.
`GracePeriod`	The approximate time to wait after start up before beginning to check whether to upgrade links. The `GracePeriod` allows time for other AMPS instances to start up, and for connections to be established between AMPS instances.

These modules do not add any variables to the AMPS context.

Extract Values

The `amps-action-do-extract-values` module extracts message values from a message and stores the values in a variable.

To extract values from a message, this module requires the `MessageType`, `Data`, and `Value` parameters.

Table 14.39. Parameters for Extract Values

Parameter	Description
`MessageType`	The MessageType for the message to parse. There is no default for this parameter.
`Data`	Contains the data to parse: typically a message received from a publish event or retrieved from a SOW query. There is no default value for this parameter. If it is omitted, AMPS will not parse data when the action is run.
`Value`	An assignment statement that specifies the variable to store the extracted value in and the XPath identifier for the value to extract. This action can contain any number of `Value` elements, each providing an assignment statement. The format of the assignment statement is as follows: `variable = amps expression` For example, the following assignment statement stores the value of the `/previousRegionCode` within the message to the variable `PREVIOUS_REGION`. After

Parameter	Description

	this action runs, the content of the variable can be referenced in subsequent actions as `{{PREVIOUS_REGION}}`.
	
	`PREVIOUS_REGION=/previousRegionCode`
	
	Likewise, the following assignment statement creates a string from the values of the `/firstName` and `/lastName` fields within the message, and stores that to the variable `COMBINED_NAME`. After this action runs, the content of the variable can be referenced in subsequent actions as `{{COMBINED_NAME}}`.
	
	`COMBINED_NAME=CONCAT(/firstName, ' ', /lastName)`
	
	There is no default for this option. If no `Value` options are provided, AMPS does not save any values from the parsed message.

The module `amps-action-do-extract-values` adds the variables specified by the `Value` options to the current context.

Translate Data

The `amps-action-do-translate-data` action allows you to translate the value from variables in the current context. One common use for this action is to translate a large number of status values into a smaller number of states before publishing that information in a message. For example, an order processing system may track a large number of finely-grained status codes, while the reporting view for customers may want to map those status codes to a smaller set of codes such as "pending", "shipped", and "delivered". This action allows you to easily translate those codes within AMPS.

When used to assemble a message, this action provides equivalent results to a set of nested conditional statements in a view projection. However, if you are using actions to parse, assemble, and publish messages, this action gives you the ability to change values.

Table 14.40. Parameters for Translate Data

Parameter	Description
`Data`	The data to translate. Most often, this is the value of a variable in the current context.
`Value`	The variable to store the translated value in.
`Case`	An translation statement. The translation statement takes the form of ***`original_value`***`=`***`translated_value`***. This action allows you to provide any number of `Case` statements. The action matches the `Data` provided to the ***`original_value`*** in each `Case` statement. When it finds a matching value, the action stores the translated value in the variable identified by the `Value` statement. For example, the following translation statement translates a value of `credit_check_in_progress` to a value of `pending` `<Case>credit_check_in_progress=pending</Case>` There is no default for this option.

Parameter	Description
`Default`	The default translation. AMPS sets the value of the variable to the contents of this element if no Case statement matches the Data provided. This element is optional. If no `Default` is specified, AMPS uses the value of the original `Data` as the default translation.

Increment Counter

The `amps-action-do-increment-counter` module allows AMPS to increment a counter by a value. Counters persist across action runs, and are saved in the instance memory until the instance is restarted.

If a counter with the specified name does not currently exist in the instance when the action runs, AMPS creates the counter with a value of 0 and then immediately increments it with the specified value. If the counter is already present, AMPS will simply increment the counter.

To see an example of `amps-action-do-increment-counter`, refer to the Action Configuration Examples section at the end of this chapter.

This module requires a `Key` that tells AMPS which counter to increment and a `Value` that tells AMPS where to store the incremented value.

Table 14.41. Parameter for Increment Counter

Parameter	Description
`Key`	The name of the counter that AMPS will increment. There is no default value for this parameter.
`Value`	The variable in which to store the current value of the counter.

This module adds variable that contains the counter, as specified in the `Value` parameter, to the current context.

Executing System Commands

The `amps-action-do-execute-system` module allows AMPS to execute system commands.

The parameter for this module is simply the command. The command executes in the current working directory of the AMPS process, with the credentials and environment of the AMPS process.

Table 14.42. Parameter for Execute System

Parameter	Description
`Command`	The command to execute. When the action runs, this command is executed as a shell command on the system where AMPS is running.

This module does not add any variables to the AMPS context.

⚠ This module executes system commands with the credentials of the AMPS process. It is possible to damage the system, interrupt the AMPS service, or cause data loss by executing commands with this module. 60East recommends against using any data extracted from an AMPS message in the command executed.

Debugging Actions

AMPS provides modules for debugging your AMPS action configuration.

Table 14.43. Debugging Modules

Module Name	Does
`amps-action-do-nothing`	Takes no action. Does not modify the state of AMPS in any way. The module simply logs that it was called.
`amps-action-do-echo-message`	Echoes the specified message to the log. The message appears in the log as message `29-0103`,at `info` level.The logging configuration must allow this message to be recorded for the output of this action to appear in the log.

The `amps-action-do-nothing` module requires no parameters.

The `amps-action-do-echo-message` module requires the following parameter:

Table 14.44. Parameter for Echo Message

Parameter	Descsription
`Message`	The message to echo. The default for this parameter is simply an empty string.

These modules do not add any variables to the AMPS context.

Creating a Minidump

AMPS provides a module for creating a minidumps. The `amps-action-do-minidump` module provides a way for developers and/or administrators to easily create minidumps for diagnostic purposes.

Table 14.45. Creating a Minidump Module

Module Name	Does
`amps-action-do-minidump`	Creates a minidump.

This module does not require any parameters.

This module does not add any variables to the AMPS context.

Shut Down AMPS

The `amps-action-do-shutdown` module shuts down AMPS. This module is registered as the default action for several Linux signals, as described in the section called “Default Signal Actions”.

Table 14.46. Shut Down Module

Module Name	Does
`amps-action-do-shutdown`	Shuts down AMPS.

This module does not require any parameters.

This module does not add any variables to the the AMPS context.

14.3. Conditionally Run Actions

AMPS includes the ability to run actions only if certain conditions are true. For some actions (such as the replication management actions), the condition is included as a part of the action. In other cases, AMPS provides `If` actions.

An `If` action is evaluated each time the execution of an action reaches the `If` action. When the condition specified in an `If` action is `true`, AMPS proceeds to the next `Do` action. If the condition in an `If` action is `False`, AMPS does not run any further `Do` elements in the action.

File System Usage

AMPS provides the following `If` module for taking action based on the file system capacity. AMPS loads this module by default:

Table 14.47. File System Usage

Module Name	Does
`amps-action-if-file-system-usage`	Checks whether the specified path on the filesystem meets the specified usage level. If so, allows execution to continue. If not, stops the action.

Table 14.48. Parameters for Running Actions Based on File System Usage

Parameter	Description
`Path`	Specifies the filesystem path to monitor. The AMPS process must have sufficient permissions to check the disk usage for this path at the time the check runs. There is no default for this parameter.
`GreaterThan`	The threshold to check, specified as a percentage. If the provided path has more space used than specified in this parameter, subsequent `Do` and `If` blocks will run. Otherwise, the action will complete with this step.

This module does not add any variables to the AMPS context.

For example, the following action will log a message in the AMPS log every minute when the file system becomes more than 90% full, and perform a full shutdown of AMPS if the file system is more than 98% full.

```
<Actions>
  <Action>
    <On>
      <Module>amps-action-on-schedule</Module>
      <Options>
         <Every>1m</Every>
      </Options>
    </On>
    <If>
      <Module>amps-action-if-file-system-usage</Module>
      <Options>
        <GreaterThan>90%</GreaterThan>
        <Path>/mnt/fastdrive/amps</Path>
      </Options>
    </If>
    <Do>
      <Module>amps-action-do-echo-message</Module>
      <Options>
        <Message>ALERT: You're getting low on space!</Message>
      </Options>
    </Do>
    <If>
      <Module>amps-action-if-file-system-usage</Module>
      <Options>
        <GreaterThan>98%</GreaterThan>
        <Path>/mnt/fastdrive/amps</Path>
      </Options>
    </If>
    <Do>
      <Module>amps-action-do-echo-message</Module>
      <Options>
        <Message>CRITICAL: Shutting down AMPS</Message>
      </Options>
    </Do>
    <Do>
      <Module>amps-action-do-shutdown</Module>
    </Do>
  </Action>
</Actions>
```

14.4. Action Configuration Examples

Archive Files Older Than One Week, Every Saturday

The listing below asks AMPS to archive files older than 1 week, every Saturday at 12:30 AM:

```
<Actions>
      <Action>
        <On>
          <Module>amps-action-on-schedule</Module>
          <Options>
            <Every>Saturday at 00:30</Every>
            <Name>Saturday Night Fever</Name>
          </Options>
        </On>
        <Do>
          <Module>amps-action-do-archive-journal</Module>
          <Options>
              <Age>7d</Age>
          </Options>
        </Do>
      </Action>
   </Actions>
```

Disable and Re-enable Security on Signal

The listing below disables authentication and entitlement when AMPS receives on the `USR1` signal. When AMPS receives the `USR2` signal, AMPS re-enables authentication and entitlement. This configuration is, in effect, the configuration that AMPS installs by default for these signals:

```
<Actions>
      <Action>
        <On>
          <Module>amps-action-on-signal</Module>
          <Options>
            <Signal>SIGUSR1</Signal>
          </Options>
        </On>
        <Do>
          <Module>amps-action-do-disable-authentication</Module>
        </Do>
        <Do>
          <Module>amps-action-do-disable-entitlement</Module>
        </Do>
      </Action>
      <Action>
        <On>
          <Module>amps-action-on-signal</Module>
          <Options>
            <Signal>SIGUSR2</Signal>
          </Options>
        </On>
        <Do>
          <Module>amps-action-do-enable-authentication</Module>
        </Do>
        <Do>
          <Module>amps-action-do-enable-entitlement</Module>
        </Do>
      </Action>
   </Actions>
```

Extract Values on Publish of a Message

The listing below extracts values from a locally published xml message and stores them into VALUE.

```
<Actions>
  <Action>
    <On>
      <Module>amps-action-on-publish-message</Module>
      <Options>
        <Topic>message-sow</Topic>
        <MessageType>xml</MessageType>
        <MessageSource>local</MessageSource>
      </Options>
    </On>
    <Do>
      <Module>amps-action-do-extract-values</Module>
      <Options>
        <MessageType>xml</MessageType>
        <Data>{{AMPS_DATA}}</Data>
        <Value>VALUE = /VALUE</Value>
      </Options>
    </Do>
  </Action>
</Actions>
```

Increment a Counter and Echo a Message on Signal

The listing below increments a counter and echoes the counter's value when AMPS receives on the `USR1` signal.

```
<Actions>
 <Action>
   <On>
     <Module>amps-action-on-signal</Module>
     <Options>
       <Signal>SIGUSR1</Signal>
     </Options>
   </On>
   <Do>
     <Module>amps-action-do-increment-counter</Module>
     <Options>
       <Key>MY_COUNTER</Key>
       <Value>CURRENT_COUNTER_VALUE</Value>
     </Options>
   </Do>
   <Do>
     <Module>amps-action-do-echo-message</Module>
     <Options>
       <Message>AMPS has gotten {{CURRENT_COUNTER_VALUE}}
                SIGUSR1 signals.</Message>
     </Options>
    </Do>
 </Action>
</Actions>
```

Copy a Message to a Different Topic When a Timeout is Exceeded

The listing below, in effect, copies messages from the `Orders` topic to the `Orders_Stale` topic when the status has been `PENDING` for more than 5 seconds.

```
<Actions>
```

```
  <Action>
    <On>
      <Module>amps-action-on-message-condition-timeout</Module>
      <Options>
        <MessageType>nvfix</MessageType>
        <Topic>Orders</Topic>
        <Filter>/status = 'PENDING'</Filter>
        <Duration>5s</Duration>
      </Options>
    </On>
    <Do>
      <Module>amps-action-do-publish-message</Module>
      <Options>
        <MessageType>nvfix</MessageType>
        <Topic>Orders_Stale</Topic>
        <Data>{{AMPS_DATA}}</Data>
      </Options>
    </Do>
  </Action>
</Actions>
```

Recording Expired Queue Messages in a Dead Letter Topic

The listing below detects when a message expires from a queue, and publishes those messages to a dead letter topic.

```
 <Action>
     <Action>
         <On>
             <Module>amps-action-on-sow-expire-message</Module>
             <Options>
               <Topic>interesting-queue</Topic>
               <MessageType>json</MessageType>
             </Options>
         </On>
         <On>
             <Module>amps-action-on-sow-expire-message</Module>
             <Options>
               <Topic>another-interesting-queue</Topic>
               <MessageType>json</MessageType>
             </Options>
         </On>
         <Do>
            <Module>amps-action-do-publish-message</Module>
            <Options>
                <Topic>dead-letter</Topic>
                <MessageType>json</MessageType>
                <Data>{"topic":{{AMPS_TOPIC}},"message":{{AMPS_DATA}} }</
Data>
            </Options>
         </Do>
      </Action>
</Actions>
```

Shutting Down AMPS When Filesystem Fills

The listing below directs AMPS to perform a graceful shutdown when the filesystem becomes full, with a check run every 3 seconds.

```
<Actions>
     <Action>
       <On>
          <Module>amps-action-on-schedule</Module>
          <Options>
           <Every>3s</Every>
          </Options>
       </On>
       <If>
         <Module>amps-action-if-file-system-usage</Module>
          <Options>
           <Path>./</Path>
           <GreaterThan>99%</GreaterThan>
         </Options>
       </If>
       <Do>
         <Module>amps-action-do-shutdown</Module>
       </Do>
     </Action>
</Actions>
```

Appendix A. Obsolete Configuration Parameters

This section lists obsolete configuration parameters and provides pointers to the current parameters, where applicable.

Table A.1. Obsolete Parameters

Element	Description
`ClientOffline`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter. To disable slow client offlining, set `MessageMemoryLimit` to 100%.
`ClientBufferThreshold`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter. To set a limit for an individual client, set `ClientMaxCapacity` to the appropriate percentage.
`ClientMaxBufferThreshold`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter. To set a limit for an individual client, set `ClientMaxCapacity` to the appropriate percentage.
`ClientOfflineThreshold`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter: offlining occurs when the total amount of space uses for messages exceeds the `MessageMemoryLimit`.
`ClientOfflineDirectory`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter. Use `MessageDiskPath` instead.
`IncrementSize`	Beginning with AMPS 5.0, SOW configuration no longer uses `IncrementSize`. Rather than setting `RecordSize` and `IncrementSize`, use the `SlabSize` parameter (to convert files, use `RecordSize` * `IncrementSize` for the `SlabSize` parameter)
`RecordSize`	Beginning with AMPS 5.0, SOW configuration no longer uses `RecordSize`. Rather than setting `RecordSize` and `IncrementSize`, use the `SlabSize` parameter (to convert files, use `RecordSize` * `IncrementSize` for the `SlabSize` parameter)
`SlowClientDisconnect`	Beginning with AMPS 5.0, slow client offlining no longer uses this parameter. To disable SlowClientDisconnect, set a large value for `MessageMemoryLimit` or `MessageDiskLimit` (for example, setting 100% of the MessageMemoryLimit will allow unrestricted memory growth, while setting the capacity of the device as MessageDiskLimit will allow unrestricted offlining).

Index