AMPS Multi-site Replication

60East Technologies

Jun 26, 2017

1. Overview

One of the powerful features of AMPS is its ability to replicate messages it receives to other instances. Replication in AMPS uses the combination of a transaction log to keep track of incoming messages and a replication transport to send the messages to downstream targets. This can provide several benefits, such as fail-over and balancing subscription loads. It can also be used to replicate data between multiple sites of a company, so that every location can have local copies of all of the data. This white paper will describe how to set up AMPS version 5.x for this scenario.

This paper presents a brief overview and describes how to configure AMPS replication for multi-site replication. For an in-depth discussion of AMPS replication, see the *AMPS User Guide*.

2. Topics and Content Filters

AMPS replication occurs only for messages matching a specified topic. All messages in an AMPS system have a topic. The topic is the highest level of message organization, and transaction logs can log all messages or only messages on specific topics. AMPS also provides high-performance content filtering in addition to topics. Replication must have a topic specified, and can also specify a content filter to further refine the data that is replicated.

3. Transaction Log

In order to maintain consistency with replicas, AMPS can only replicate messages that go into the transaction log on both sides of the replication connection. AMPS acknowledges receipt of a message after the message is saved in the transaction log. This allows AMPS to recover from failure and request replay from the correct point in the message flow.

4. Basic Replication

Replication is the term that describes having one AMPS instance send messages it receives from publishers to another AMPS instance. We will refer to the instance that receives messages from the original publisher as the **main instance**, and the instance that receives the replicated messages as the **target instance**.

AMPS uses a special replication transport to forward messages to another AMPS instance. By default, messages received by the target instance are not forwarded. Since replicated messages only travel over one downstream link by default (to prevent looping), you can configure instances to replicate to each other without worrying about causing replication loops.

AMPS replication is always asynchronous from the point of view of subscribers. That is, AMPS publishes a message to subscribers as soon as the message is recorded in the local transaction log. For publishers, AMPS supports two acknowledgement modes. With the *synchonous* acknowledgement mode, all downstream instances must have persisted the message before it is acknowledged as persisted to the publisher. With the *asynchronous* acknowledgement mode, the message is acknowledged to a publisher as soon as it is persisted in the local transaction log. It's important to keep in mind that the difference between synchronous acknowledgement and asychronous acknowledgement is only relevant for the publisher of the message: there is no difference in how the message is replicated or when it is available to subscribers.

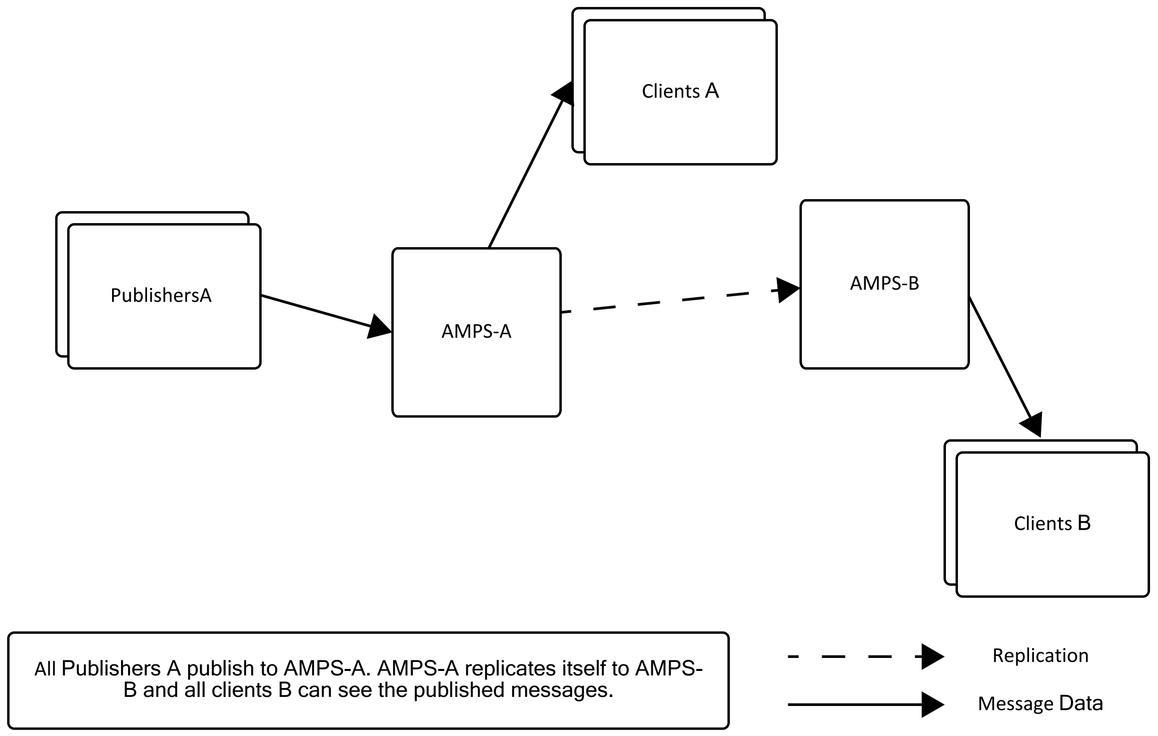
Clients A
PublishersA
AMPS-A
AMPS-B
Clients B
All Publishers A publish to AMPS-A. AMPS-A replicates itself to AMPS-B and all clients B can see the published messages.
Replication
Message Data

Figure 1. Basic Replication

5. Multi-site Replication

Beyond basic replication, AMPS includes features designed for replication between sites. The features used in this paper are:

- **Passthrough replication**. As described above, by default replication sends messages from the main instance to the target instance and no further. With *passthrough replication*, a target instance will replicate messages received from the main instance if the main instance is in a specific group. This allows a multi-site AMPS installation to simplify configuration: if a message reaches a single instance at the remote site, that instance can guarantee that the message is delivered to the other instances at the site.

- **Failover destination lists**. AMPS allows you to specify a set of network addresses for a single destination. This provides the ability for a replication link to fail over to another server in the same group, which means that replication can continue even if the the preferred destination is unreachable or offline. When a failover list is provided, AMPS connects to only one server in the list.

- **Replication compression**. AMPS supports transparent replication compression to reduce the bandwidth required for replication.

These features are all available in AMPS 4.3 and later releases. Usage of these features in the sample scenario is described below.

In AMPS 5.0 and later releases, AMPS adds the ability to automatically validate that instances have compatible configuration files. Also in 5.0 and later releases, AMPS can use a single network connection between instances for replication: in cases where instances have bidirectional replication, AMPS can optimize the network connections by replicating both directions over a single connection. The scenario described in this whitepaper doesn't require either of these features, but those features can make it easier to configure and manage AMPS replication deployments.

6. Scenario for Replication

This document uses the following scenario for implementing multi-site replication. JKL Corp is currently using AMPS at each location (New York, Chicago, and London) to publish some usage, error, and hardware data for their IT departments. All three sites use the same topics and format for their data. All errors are published on the topic `/it_error`. The local departments would like to have a world-wide view so they can quickly determine if problems are local or related to some globally-deployed solution. Because they already use AMPS to pass around this custom data internally, AMPS is the logical choice for sharing the data globally.

In this scenario, we have three AMPS instances that we want to replicate. Note that AMPS instances can replicate to 64 synchronous destinations and any number of asynchronous destinations, but administrators must be mindful of their resource limits in setting up replication. We will replicate each instance to the other two instances. We need to configure this keeping in mind that the replications are happening to remote locations, so we dont want them slowing down local processing. Figure 2 provides the layout for this configuration.

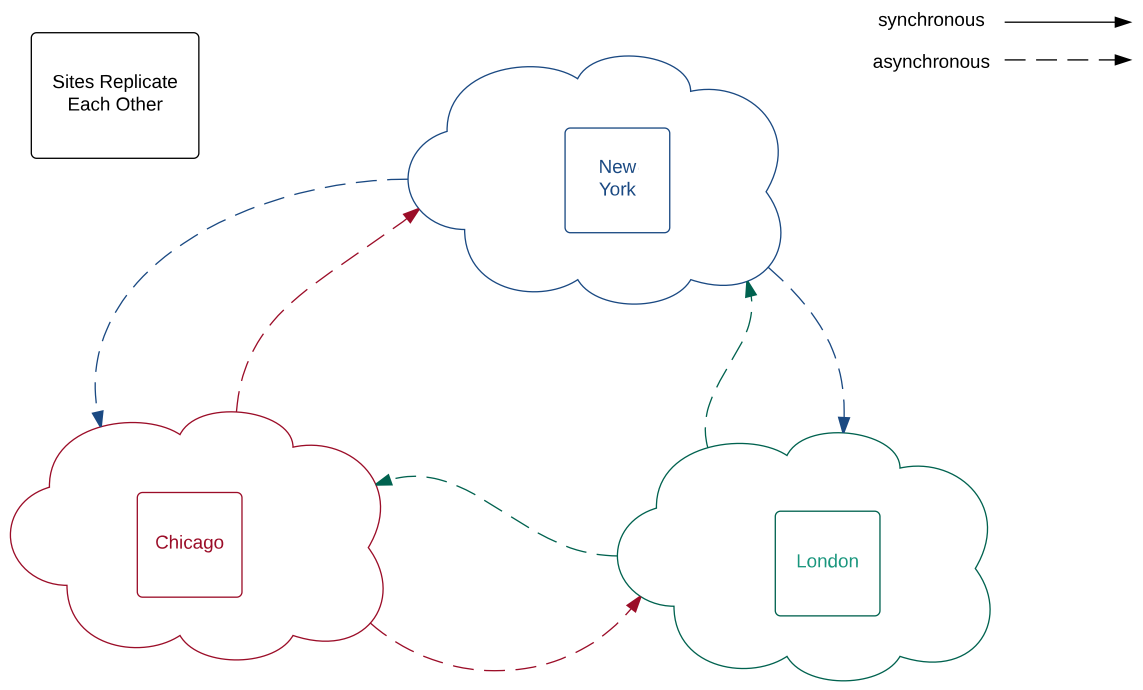

Figure 2. AMPS Multi-site replication for JKL corp.

As you can see from the figure, each site initiates replication to the other two sites. We will set up the replication to be asynchronous. Publishers will continue to publish to their known AMPS instance. Clients will continue to subscribe to the same AMPS instance, but will now also be able to see messages from the other AMPS instances matching their subscriptions. A client can connect to any AMPS instance to see the data from all sites.

Here, we use asynchronous acknowledgements so that publishers do not need to retain messages until they are replicated to all sites. Once the local AMPS instance has stored the message to the local transaction log, that instance will acknowledge the message to the publisher. This is a common pattern for multi-site replication: allow acknowledgement when the message is persisted at the local site by using asynchronous acknowledgements on replication to the remote sites.

7. Configuration File

The configuration files for each of the instances in our scenario can be nearly identical. The only place where they need to differ is in the names of the instances and the names of the replication targets. Each configuration file needs to have a `<TransactionLog>` for the replicated topic. The following configuration block adds a transaction log for the `/it_error` topic.

```
<TransactionLog>
  <JournalDirectory>./amps/journal</JournalDirectory>
  <MinJournalFiles>2</MinJournalFiles>
  <MinJournalSize>10MB</MinJournalSize>
  <BatchSize>128</BatchSize>
  <Topic>
    <Name>/it_error</Name>
    <MessageType>json</MessageType>
  </Topic>
</TransactionLog>
```

Example 1. Transaction Log Configuration

Each server also needs a `<Replication>` section in the configuration file. We need to configure two destinations for each server. The below segment is used in London, and with minor changes at the other locations.

```
<Replication>
  <Destination>
      <Group>Chicago</Group>
      <SyncType>async</SyncType>
      <Compression>enabled</Compression>
      <Transport>
          <Name>amps-replication</Name>
          <Type>amps-replication</Type>
          <InetAddr>chicago.jkl.com:9290</InetAddr>
          <ReuseAddr>true</ReuseAddr>
      </Transport>
      <Topic>
          <Name>/it_error</Name>
          <MessageType>json</MessageType>
      </Topic>
  </Destination>
  <Destination>
      <Group>NewYork</Group>
      <SyncType>async</SyncType>
      <Compression>enabled</Compression>
      <Transport>
          <Name>amps-replication</Name>
          <Type>amps-replication</Type>
          <InetAddr>newyork.jkl.com:9290</InetAddr>
          <ReuseAddr>true</ReuseAddr>
      </Transport>
      <Topic>
          <Name>/it_error</Name>
          <MessageType>json</MessageType>
      </Topic>
  </Destination>
</Replication>
```

Each destination is configured with `<SyncType>` set to `async`. This allows AMPS to acknowledge receipt of the message to the publisher before receiving an acknowledgement of receipt from the replication destination. The `<Name>` for each destination should be unique, but can be whatever best suits the user. The `<Topic>` must match a topic that is being written to a transaction log. JKL Corp could further refine the replication by adding a `<Filter>` to the `<Topic>` section of the destination. Because the instances are at different locations, bandwidth is at a premium, so we specify that the connections will enable `<Compression>` to preserve bandwidth.

Because each `Destination` is replicating to multiple servers in the same `Group`, we set the `Group` for the `Destination` to the remote `Group`. Since we only provide one destination to each `Group`, there's no need to set a distinct `Name` in the `Destination` -- letting the `Name` default to the `Group` will work perfectly, and slightly simplify the configuration.

All replication destinations need to set up a `<Transport>` for replication in their configuration file. This transport will be listening for replication messages. The `<InetAddr>` must match what was in the `<Replication>` section within the config files for the other two instances. As shown below, we add the `<InetAddr>` that will receive incoming replication connections to the `<Transports>` section of the configuration file.

```
<Transports>

  <!-- transports for client use -->

  ...

  <!-- listen to local port 9290
       for incoming replication
       connections             -->

  <Transport>
    <Name>amps-replication</Name>
    <Type>amps-replication</Type>
    <InetAddr>9290</InetAddr>
    <ReuseAddr>true</ReuseAddr>
  </Transport>

</Transports>
```

As with connections for clients, any number of remote instances can connect to the same incoming `Transport`. In most cases, there is no need for more than one replication `Transport`. If we need this instance to replicate to other downstream instances later, we will update the configuration file for each instance by adding new destinations to the `<Replication>` section of the configuration file. Each new destination will have a configuration similar to the existing configuration.

8. Adding High Availability

In the configuration described above, the AMPS instance is able to recover from a failure. The transaction log will allow the instance to know where to begin replicating messages. Assuming that publishers are also configured for high availability (see *AMPS User Guide*, "Publishing for High Availability" and the high availability section of the relevant *Developer Guide*), an installation should be able to fully recover with no lost messages. The configuration described above provides *recoverability*, which means that publishers and clients are all down while the instance is down, but can resume without message loss when the AMPS instance recovers.

For *high availability* it is more common to use a pair of servers running "hot-hot," meaning that they are both always running. This reduces downtime further in the case of a server failure. Publishers and subscribers will need to be properly configured for a pair scenario (see *AMPS User Guide* and the high availability section of the relevant *Developer Guide*). Typically, this involves using the `HAClient` (high availability client) with an appropriate `ServerChooser` implementation.

Replication plays a key role in high availability with a pair of servers. The servers will be configured "hot-hot," but one will be considered the primary server as the target for publishers, while the secondary will take over if the primary goes down. The replication from the primary server to the secondary server should be configured with a `<SyncType>` of `sync`. This will guarantee that messages are persisted to both servers before AMPS acknowledges the message to the publisher: this provides consistency in cases where a servier temporarily loses the network connection or fails due to problems with persistent storage.

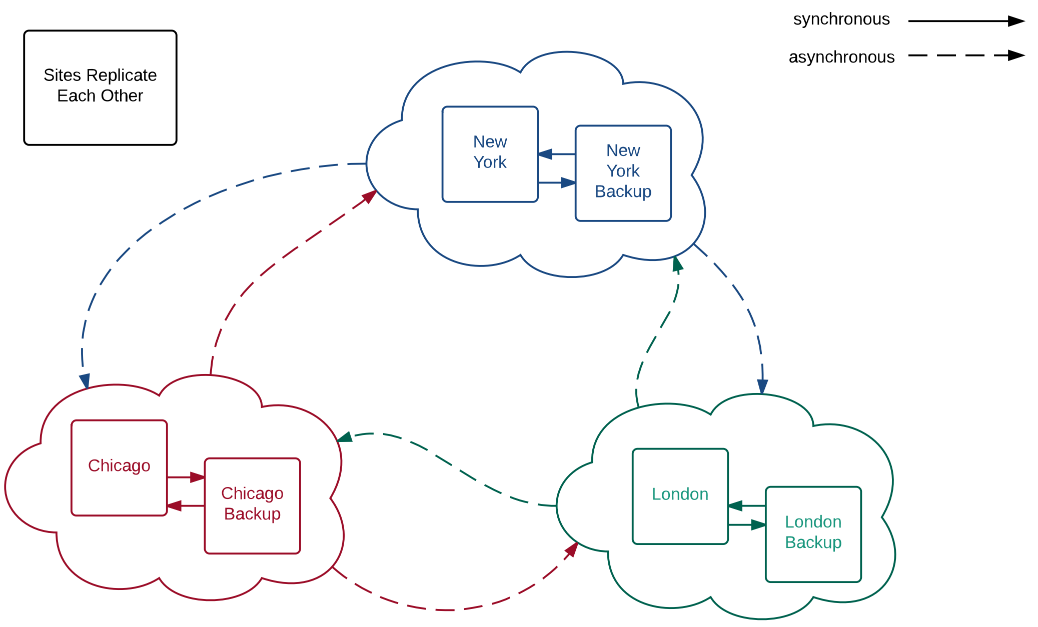

Figure 3. Multi-site replication with HA at each site

Configuring the HA Pair

To configure an HA pair, the primary server configuration will add a new replication destination. That destination will be the secondary server. The primary server will also use passthrough replication to replicate messages from the other two groups to the secondary, as shown below:

```
<Replication>
 ...
  <Destination>
    <Group>London</Group>
    <Transport>
      <Name>amps-replication</Name>
      <Type>amps-replication</Type>
      <InetAddr>london-secondary.jkl.com:9290</InetAddr>
      <ReuseAddr>true</ReuseAddr>
    </Transport>
    <Topic>
      <Name>/it_error</Name>
      <MessageType>json</MessageType>
    </Topic>
    <SyncType>sync</SyncType>
    <PassThrough>Chicago|NewYork</PassThrough>
  </Destination>
 ...
</Replication>
```

Example 2. Primary Local Replication Configuration

The `PassThrough` element means that any message received from a server in the Chicago or NewYork groups will be *pass through* this instance and be replicated to the secondary, even though that message originated with another instance.

The secondary server has the same basic replication configuration, except that it changes the `InetAddr` to replicate to the primary rather than to itself.

```
<Replication>
...
  <Destination>
    <Group>London</Group>
    <Transport>
      <Name>amps-replication</Name>
      <Type>amps-replication</Type>
      <InetAddr>london.jkl.com:9290</InetAddr>
      <ReuseAddr>true</ReuseAddr>
    </Transport>
    <Topic>
      <Name>/it_error</Name>
      <MessageType>json</MessageType>
    </Topic>
    <SyncType>sync</SyncType>
    <PassThrough>Chicago|NewYork</PassThrough>
  </Destination>
...
```

```
</Replication>
```

Example 3. Secondary Local Replication Configuration

Adding Remote Addresses

To ensure that messages reach the remote sites even when the primary server for the remote site is offline, we simply add the address for the backup server at each site to the destination for that site. The updated sample for London, below, adds alternate addresses to the Chicago and NewYork sites. Notice that replication will only connect to one `InetAddr` in each destination at a time. This means that, for each group, AMPS will connect to the primary if that is available, and connect to the secondary if the primary is unavailable. Because each of the HA pairs use `<PassThrough>`, it is only necessary to ensure that the message reaches one of the servers in the remote group. That server will distribute messages to the other server at the remote site. The figure below shows the result of the config changes.

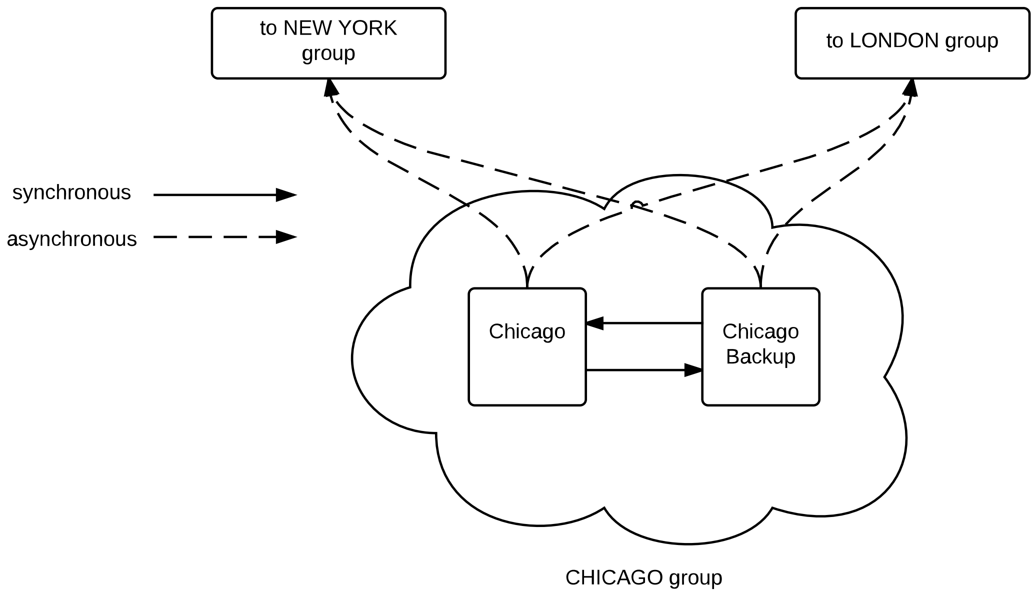

Figure 4. Detail on Replication Configuration with HA

Both servers in an HA pair use the same configuration for servers in other groups. The sample below shows the updated destinations in London replicating to the remote sites:

```
<Replication>
  ...
  <Destination>
      <Group>Chicago</Group>
      <SyncType>async</SyncType>
      <Compression>enabled</Compression>
      <Transport>
          <Name>amps-replication</Name>
          <Type>amps-replication</Type>
          <!-- Failover destination list for Chicago -->
          <InetAddr>chicago.jkl.com:9290</InetAddr>
          <InetAddr>chicago-secondary.jkl.com:9290</InetAddr>
          <ReuseAddr>true</ReuseAddr>
      </Transport>
      <Topic>
          <Name>/it_error</Name>
          <MessageType>json</MessageType>
      </Topic>
  </Destination>
  <Destination>
      <Group>NewYork</Group>
      <SyncType>async</SyncType>
      <Compression>enabled</Compression>
      <Transport>
          <Name>amps-replication</Name>
          <Type>amps-replication</Type>
          <!-- Failover destination list for New York -->
          <InetAddr>newyork.jkl.com:9290</InetAddr>
          <InetAddr>newyork-secondary.jkl.com:9290</InetAddr>
          <ReuseAddr>true</ReuseAddr>
      </Transport>
      <Topic>
          <Name>/it_error</Name>
          <MessageType>json</MessageType>
      </Topic>
  </Destination>
  ...
</Replication>
```

The only difference between this configuration and the previous, single-instance, configuration is that each destination will fail over to the secondary server in the HA pair. This ensures that replication to the remote site will continue even if one of the servers in the remote HA pair goes offline.

The strategy for passthrough is designed to provide strong reliability guarantees while minimizing the number of times a message crosses the WAN. As configured above, the passthrough strategy is as follows:

- Each instance provides passthrough for messages that arrive from remote groups. This means that a message that arrives via replication from a remote group need only arrive at one server in the local group to be replicated to the other server.

- Each instance replicates messages published *directly to* that instance (rather than received via replication) to a server in each remote group. All of the servers in a remote group are listed in a failover destination list as failover equivalents, so the message only travels to the remote group once. The message will be replicated as long as at least one instance in the remote group is available.

- Instances do NOT provide passthrough for messages via replication in the local group. This prevents an instance from replicating a message to a remote group that has already been replicated by the instance that received the original publish.

This configuration provides a high degree of confidence that messages will be replicated to every server in the topology, while minimizing the number of times an individual message is replicated.

9. Managing Replication

AMPS provides a number of administrative actions to make it easy to manage replication in high-volume installations. To manage replication, you set policy for:

- How long transaction logs will be retained on high-speed storage before being archived to slower, higher-capacity storage.
- Whether to compress archived transaction logs to save space.
- How long archived transaction logs will be retained.
- Whether it is acceptable to downgrade the connection between an HA pair if one of the servers is offline for an extended period of time. If so, how long must the connection be down before downgrading the connection. Downgrading a connection to synchronous acknowledgement to asynchronous acknowledgemnt can reduce resource requirements on publishers if a server is offline. The *AMPS User Guide* has more details on downgrading and upgrading replication links.

For each of these policies, you also decide at what interval AMPS will check the policy and take appropriate action. Once you've made these decisions, you create the appropriate actions to enforce the policies in the `Actions` section of the configuration file.

In our example installation, we will set the following policies:

- Transaction log journal files will be retained for 2 days on high speed storage.
- We will compress journal files older than 2 days.
- Archived transaction log journal files will be retained for 14 days.
- If our HA partner fails (or performance degrades), we will downgrade the connection if the connection is more than 5 minutes behind. When the connection returns (or performance improves), we will upgrade the connection when it is less than 2 minutes behind.

Now that we have chosen the policies, we need to decide how often we will check the policies.

- We will check files to be archived twice a day.
- We will check upgrade and downgrade status every 5 minutes.
- On startup, we will avoid taking any upgrade or downgrade action for 5 minutes to allow for startup and for connections to be established to the HA partner.

The following configuration block implements our chosen policies:

```
<Actions>
   <!-- POLICY: archive and compress journals older
        than 2 days, remove journals older than
```

```
     14 days. -->
<Action>
    <On>
       <Module>amps-action-on-schedule</Module>
       <Options>
          <Every>12h</Every>
          <Name>Transaction log management</Name>
       </Options>
    </On>
    <Do>
       <Module>amps-action-do-archive-journal</Module>
       <Options>
            <Age>2d</Age>
       </Options>
    </Do>
    <Do>
       <Module>amps-action-do-compress-journal</Module>
       <Options>
           <Age>2d</Age>
       </Options>
    </Do>
    <Do>
       <Module>amps-action-do-remove-journal</Module>
       <Options>
           <Age>14d</Age>
       </Options>
    </Do>
</Action>
<!-- POLICY: If a replication connection is behind
     farther than 5 minutes, downgrade it. Upgrade
     it if it catches up to being less than 2 minutes
     behind. -->
<Action>
    <On>
       <Module>amps-action-on-schedule</Module>
       <Options>
          <Every>5m</Every>
          <Name>Manage replication connections</Name>
       </Options>
    </On>
    <Do>
       <Module>amps-action-do-downgrade-replication</Module>
       <Options>
           <Age>5m</Age>
           <GracePeriod>5m</GracePeriod>
       </Options>
    </Do>
    <Do>
       <Module>amps-action-do-upgrade-replication</Module>
       <Options>
          <Age>2m</Age>
          <GracePeriod>5m</GracePeriod>
       </Options>
    </Do>
    </Action>
</Actions>
```

10. Summary

Replication and transaction logging are two of many powerful AMPS features. Replication is an important tool for both sharing information in distributed organizations and in high availability scenarios. AMPS provides message replication and transaction logging via your server configuration. The transaction log allows AMPS to provide a consistent set of data to downstream instances. AMPS uses synchronous replication in high availability situations to guarantee a constantly, consistent state between servers. It offers asynchronous replication for situations where the servers are remote. Using a combination of these features can provide you with multi-site replication and high availability at every location.

FOR MORE INFORMATION, CONTACT:

60East Technologies

www.crankuptheamps.com [http://www.crankuptheamps.com]

`support@crankuptheamps.com`

(888) 206-1365