AMPS C# EVALUATION KIT

Version 5.0, 2016-03-04

Welcome to AMPS!

AMPS, the Advanced Message Processing System, is a reliable high-performance publish and subscribe engine designed for real-world messaging applications that demand the highest performance and lowest latency possible. AMPS solves the hardest problems for modern pub/sub and queueing applications in a way that maximizes performance and reliability without compromising ease of use.

What makes AMPS different?

AMPS is designed for extremely demanding, real-world messaging applications. This means that AMPS has:

- Uncompromising focus on real-world needs
    - Hardware-level optimization for performance
    - Easy-to-use programming interface
    - Detailed monitoring and diagnostics
- Complete solution for high-volume low-latency applications
    - Durability and transactional consistency
    - Both pub/sub and queueing delivery models
    - Replication for high availability
    - Persistent state-of-the-world database
    - Content filtering with SQL92 WHERE clause semantics
    - Topic views, projection and JOIN
    - Message replay and historical point-in-time query
    - Real-time computation and analysis using standard SQL semantics
    - Delta publish and subscribe for lightweight updates
    - Built-in handling for FIX, NVFIX, JSON, BSON, Google Protocol Buffer and XML messages

AMPS allows you to configure each server, topic, or view to use exactly the features that you need. This keeps AMPS programming and administration simple.

Get the Equipment

To get the most out of your evaluation, you will need:

- A Linux installation, either on a Virtual Machine running Linux or on dedicated hardware. If you will be developing on Linux, we recommend a dedicated installation. If you will be developing on Windows, a virtual machine running Linux hosted on your Windows system is a convenient way to develop with AMPS.
    - If you do not have VMWare Player, you can download it free for personal use at: http://www.vmware.com/products/player/
    - Make sure that Virtualization Technology is enabled on your development machine, as described at http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1003944
- The AMPS Java Evaluation Kit (which this file is part of). Visit crankuptheamps.com to download the latest kits and documents.
- Download the AMPS Linux distribution at http://crankuptheamps.com/evaluation.

Turn It On

To install the Linux distribution of AMPS and start the evaluation server:

1. Extract the Linux distribution and the evaluation kit.
2. Make a config directory in the AMPS distribution directory, for example:

```
$ mkdir ~/amps_dir/config
```

3. Copy the sample.xml file from the evaluation kit into the config directory, for example:

```
$ cp ~/CppEvaluationKit/sample.xml ~/amps_dir/config/sample.xml
```

4. Change directories to the AMPS distribution directory and start AMPS with the sample config file. For example:

```
$ cd ~/amps_dir/

$ ./bin/ampServer ./config/sample.xml
```

Crank It Up!

To get started with AMPS, create a simple subscriber and publisher:

1. Open the AMPS C# samples in Visual Studio

a. Navigate to the directory where you unzipped the evaluation kit

b. Navigate to **C# Eval Kit/CrankItUp**

c. Open the **CrankItUp** solution

2. Update the connection strings to connect to AMPS

    a. In Visual Studio, choose **Edit > Find and Replace > Replace in Files** (or type **Ctrl-Shift-H**)

    b. Click on **Replace in Files**

    c. In the **Find What** text box, enter **127.0.0.1**

    d. In the **Replace With** text box, enter the IP address of the virtual machine (provided when the machine boots and when you log in to the virtual machine)

3. Run the sample subscriber

    a. In the solution view, right click on **01-AMPSConsoleSubscriber**. Select **Debug > Start new instance**.

    b. The program starts in a console window.

    c. Choose **Debug > Detach all**. You can start as many subscribers as you like this way.

4. Run the sample publisher

    a. In the solution view, right click on **02-AMPSConsolePublisher**. Select **Debug > Start new instance.**

    b. The program runs, and subscribers receive the Hello, World message. The subscriber itself produces no output if it is successful.

One of the most useful parts of AMPS is being able to keep track of the current value of a message, using the state of the world (SOW) database:

5. Add messages to a state of the world database

    a. In the solution view, right click on **03-AMPSSOWConsolePublisher**. Select **Debug > Start new instance**.

    b. The program runs. This program sends 100 messages to AMPS. In this case, the messages are sent on a topic, `messages-sow`, that maintains a state-of-the-world database. AMPS saves the most recent version of each unique message. Unique messages are identified by the `messageNumber` in the message. The program sends messages 0-99, then updates message 5.

8. Retrieve the state of the world

    a. In the solution view, right click on **04-AMPSSOWConsoleSubscriber**. Select **Debug > Start new instance**.

b. The program runs. This program queries the state of the world for the `messages-sow` topic, filtered to messages with a `messageNumber` less than 10. The console output shows the results of the query. Notice that for message 5, the database shows the latest value – the current state of the world.

9. Retrieve the state of the world and subscribe to updates.

a. In the solution view, right click on **05-AMPSSOWandSubscribeConsoleSubscriber**. Select **Debug > Start new instance**.

b. The program runs. This program queries the state of the world for the `messages-sow` topic, filtered to messages with a `messageNumber` less than 10. The console output shows the results of the query. Notice that for message 5, the database shows the latest value – the current state of the world. Rather than ending, the program continues to run. AMPS will send the program any new messages that match the filter.

c. Choose **Debug > Detach all**.

d. In the solution view, right click on **03-AMPSSOWConsolePublisher**. Select **Debug > Start new instance**.

e. The publisher sends another 100 messages to AMPS, updating the value of each message, then updates message 5 again. Notice that AMPS sends each update to the subscriber, meaning that the subscriber receives two updates to message 5. When subscribed, the subscriber always receives the most current information.

Detecting when messages go out of focus

One of the difficulties in conventional messaging systems and databases is being notified when an item no longer matches your query. For example, you may want to keep a list of all of the orders which are not yet fulfilled, or all of the taxis that are available to pick up passengers. In a traditional system, you detect change by running the query again and comparing the new results to the previous results.

With AMPS, you can ask to be notified when a message no longer matches the query. AMPS delivers the notification as a special kind of message – an Out of Focus message.

10. Retrieve the state of the world and request Out of Focus notifications.

a. In the solution view, right click on **07-AMPSSOWandSubscribeWithOOF**. Select **Debug > Start new instance**.

b. The program runs. This program queries the state of the world for the `messages-sow` topic, filtered to messages with a `messageNumber` is a multiple of 10, and where an optional field is either not present or set to a value other than `'ignore_me'`. The console output shows the results of the query.

c. Choose **Debug > Detach all**.

d. In the solution view, right click on **06-AMPSSOWUpdateForOOF**. Select **Debug > Start new instance**.

e. The program runs. This program does the following work:

- Sends a message with an expiration time. The subscriber receives the message.
- Sends a message that it will later delete. The subscriber receives the message.
- Sends a set of messages that match the filter. The subscriber receives these messages.
- Sends updates to a set of messages that cause them to no longer match the filter. The subscriber receives Out Of Focus messages. Notice that these messages include the information that the filter no longer matches.
- Deletes the message that it sent earlier. The subscriber receives an Out of Focus message saying that the message was deleted.

The program waits to help ensure that all messages have been published, then exits. The message with the expiration time set will expire, and the subscriber will receive an Out of Focus message saying that the message expired.

Go beyond

The sample directory contains a number of sample programs, including programs that demonstrate publishing to and consuming from AMPS queues, programs that work with composite messages, and more.

Using the spark command line tool

The evaluation kit includes `spark`, a command line tool for working with AMPS. You can use `spark` to quickly test commands and see the results in AMPS. Here are some quick examples to get you started:

Check to see if an AMPS server is reachable at the specified address

```
$ ./spark ping -type fix -server localhost:9004
```

Publish from standard input:

```
$ echo '<hi><text>Hello,
Spark</text><messageNumber>42</messageNumber></hi>' | ./spark
publish -type xml -server localhost:9006 -topic messages-sow
```

Retrieve the current state of a SOW topic:

```
$  ./spark sow -type xml -server localhost:9006 -topic messages-
sow
```

Retrieve SOW messages matching a filter:

```
$ ./spark sow -type xml -server localhost:9006 -topic messages-
sow -filter '/hi/messageNumber % 23 = 0'
```

SOW and Subscribe:

```
$ ./spark sow_and_subscribe -type xml -server localhost:9006 -
topic messages-sow -filter '/hi/messageNumber in (1, 5, 7, 42) OR
/hi/messageNumber > 90'
```

You call the tune

Now that you've seen the samples run, it's time for you to play with the samples. Modify them. Step through them in a debugger to see how they work.

Faster. Louder. Better.

From here, the world of AMPS is yours to explore. The Evaluation Kit gives you a set of guides to help you understand how AMPS works and what you need to do to take advantage of AMPS in your applications.

The *AMPS Evaluation Guide* orients you to the capabilities of the system. The *AMPS User Guide* describes the features of AMPS in detail, with a focus on helping you understand the concepts behind AMPS. The *AMPS C++ Developer Guide* describes the C and C++ client interface for AMPS.

You know what your application needs. We've provided some suggestions for simple programs that will help you get familiar with AMPS and help you understand how AMPS can help you.

Suggested learning projects:

- *Understanding the state of the world*. Fill the SOW database with messages. Verify this using the `amps_sow_dump` tool. Update a few messages in the database. Verify the updates using the `amps_sow_dump` tool. Then write a program that uses SOW and delta subscriptions to get the same information.

- *Working with multiple AMPS instances*. Create an application that forwards selected messages from one AMPS instance to another.

- *Working with views*. Create a view in AMPS that does simple calculation on values within incoming messages. Create an application that uses a filtered subscription on the view to view messages where the calculated value is within a certain range. Modify the application to use an out-of-focus message to be notified when the calculated value is outside of that range.

Keep It Going

Want to know more about how AMPS can help you build great applications? Still having trouble getting AMPS to play your tune? For help or questions, send us a note at support@crankuptheamps.com. And if AMPS ever overloads (crashes with a minidump), send the dump file to crash@crankuptheamps.com.