IBM MQ for z/OS V9.1.3

AMS Interception on Server to Server Message Channels

Version 1.0 – July 2019

Tony Sharkey

IBM MQ Performance
IBM UK Laboratories
Hursley Park
Winchester
Hampshire

Notices

DISCLAIMERS
The performance data contained in this report was measured in a controlled environment. Results obtained in other environments may vary significantly.

You should not assume that the information contained in this report has been submitted to any formal testing by IBM.

Any use of this information and implementation of any of the techniques are the responsibility of the licensed user. Much depends upon the ability of the licensed user to evaluate the data and to project the results into their own operational environment.

WARRANTY AND LIABILITY EXCLUSION

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

ERRORS AND OMISSIONS
The information set forth in this report could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; any such change will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time and without notice.

INTENDED AUDIENCE
This report is intended for architects, systems programmers, analysts and programmers wanting to understand the performance characteristics of *IBM MQ V9.1*. The information is not intended as the specification of any programming interface that is provided by IBM MQ. It is assumed that the reader is familiar with the concepts and operation of IBM MQ V9.1.

LOCAL AVAILABILITY
References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

ALTERNATIVE PRODUCTS AND SERVICES
Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

USE OF INFORMATION PROVIDED BY YOU
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

TRADEMARKS AND SERVICE MARKS
The following terms used in this publication are trademarks of their respective companies in the United States, other countries or both:

- **IBM Corporation:** IBM

Other company, product, and service names may be trademarks or service marks of others.

EXPORT REGULATIONS
You agree to comply with all applicable export and import laws and regulations.

Table of Contents

Introduction

IBM MQ for z/OS version 9.1.3 Continuous Delivery Release (CDR) introduces support for AMS Interception on Server to Server Message Channels.

This paper will talk about what AMS Interception on Server to Server Message Channels is, how to configure your MQ for z/OS systems to use this new function, and what impact you may see to the performance of your AMS-enabled queue manager.

AMS Interception on Server to Server Message Channels is not the same, and should not be confused with AMS MCA Interception, which is a distributed only feature and can be used to selectively enable policies to be applied for server connection-type channels.

For the purposes of this document, we refer to AMS Interception on Server to Server Message Channels as “AMS Interception on z/OS”.

The IBM MQ Knowledge Centre should be regarded as the primary repository for information on configuring MQ for z/OS with AMS.

Background

The Advanced Message Security component of IBM MQ for z/OS provides a high level of protection for sensitive data flowing through the MQ network with different levels of protection by using a public key cryptography model.

IBM MQ for z/OS offers three qualities of protection *Integrity, Privacy* and *Confidentiality*.

Integrity protection is provided by digital signing, which provides assurance on who created the message, and the message has not been altered or tampered with.

Privacy protection is provided by a combination of digital signing and encryption. Encryption ensures that message data is viewable by only the intended recipient, or recipients.

Confidentiality protection is provided by encryption only.

AMS uses a combination of symmetric and asymmetric cryptographic routines to provide digital signing and encryption. Asymmetric key operations are costly, but may benefit from offloading to cryptographic hardware such as Crypto Express 6S. In comparison, symmetric key operations are very fast but are not eligible for offload to the cryptographic hardware.

- Asymmetric cryptographic routines as used by *Integrity* and *Privacy*

For example, when putting a signed message the message hash is signed using an asymmetric key operation. When getting a signed message, a further asymmetric key operation is used to verify the signed hash. Therefore, a minimum of two asymmetric key operations are required per message to sign and verify the message data. Some of this asymmetric cryptographic work can be offloaded to cryptographic hardware.

- Asymmetric and symmetric cryptographic routines as used by *Privacy* and *Confidentiality*

When putting an encrypted message, a symmetric key is generated and then encrypted using an asymmetric key operation for each intended recipient of the message. The message data is then encrypted with the symmetric key. When getting the encrypted message the intended recipient needs to use an asymmetric key operation to discover the symmetric key in use for the message. The symmetric key work cannot be offloaded to cryptographic hardware but will be performed in part by CPACF processors.

All three qualities of protection, therefore, contain varying elements of CPU intensive asymmetric key operations, which will significantly impact the maximum achievable messaging rate for applications putting and getting messages.

Confidentiality policies do, however, allow for symmetric key reuse over a sequence of messages.

Key reuse can significantly reduce the costs involved in encrypting a number of messages intended for the same recipient or recipients.

For example, when putting 10 encrypted messages to the same set of recipients, a symmetric key is generated. This key is encrypted for the first message using an asymmetric key operation for each of the intended recipients of the message.

Based upon policy controlled limits, the encrypted symmetric key can then be reused by subsequent messages that are intended for the same recipient(s). An application that is getting encrypted messages can apply the same optimization, in that the application can detect when a symmetric key has not changed and avoid the expense of retrieving the symmetric key.

In this example, 90% of the asymmetric key operations can be avoided by both the putting and getting applications reusing the same key.

What is AMS Interception on z/OS and when would I use it?

AMS Interception on z/OS provides a means to control whether messages should have any applicable AMS policies applied to them, when sender-type message channel agents get messages from transmission queues, and receiver-type message channel agents put messages to target queues.

This allows AMS protection to be enabled on a queue manager when communicating using server to server channels of type sender, server, receiver and requester, with a queue manager that does not have AMS enabled.

In turn, this means that AMS protected messages in AMS protected queue managers can be unprotected prior to being sent to non-AMS enabled queue managers, and unprotected messages received from non-AMS enabled queue managers can be protected, by applicable AMS policies, on AMS enabled queue managers.

AMS Interception on z/OS allows the enterprise to protect their data using AMS qualities of protection without mandating all of their business partners apply AMS protection or even the same AMS quality of protection.

AMS interception on z/OS can also be used to ensure data hosted by discrete '[data islands]' within the same business can protect their local data, without needing to co-ordinate the certificate management. The use of TLS cipher protection over MQ channels may be sufficient to protect the data in-flight between those islands, based on the business' security policies, whilst relying on AMS to provide protection of data at rest on the island.

Note that AMS is not a replacement for using TLS on channels, however for simplicities sake, the measurements in this document do not use TLS cipher protection on the MQ channels.

Given the above use cases for AMS Interception on z/OS, it is anticipated that the typical configuration will use the AMS Confidentiality quality of protection.

As such, the measurements in this document will focus primarily on relying on Confidentiality to encrypt the data, although examples with Integrity and Privacy are also provided.

Terminology used in this document

Given the scope of the AMS Interception on z/OS feature, all workloads will be queue manager to queue manager using MQ server to server channels. Each workload has a domain where the data is protected at a similar level of protection. We are using the term “data island” to represent these areas of data that are protected in the same manner.

This document will consider scenarios of the following types:

- “Single data island” across 2 queue managers, with unprotected workload.
- “Single data island” across 2 queue managers, with end-to-end AMS protection.
- “Multiple data islands”, one protected with AMS Interception on z/OS and one unprotected.
- “Multiple data islands”, protected independently with AMS Interception on z/OS.

Configuring AMS Interception on z/OS

AMS Interception on z/OS is configured using the **SPLPROT** attribute on channel types (CHLTYPE) of SDR, SVR, RCVR or RQSTR. The available options to configure the behavior are dependent on the channel type specified:

PASSTHRU
Pass through, unchanged, any messages sent or received by the message channel agent for this channel. This value is valid for channels with a channel type (CHLTYPE) of SDR, SVR, RCVR or RQSTR and is the default value.

REMOVE
Remove any AMS protection from the message retrieved from the transmission queue by the Message Channel Agent (MCA), and send the messages to the remote partner.
When the MCA gets a message from the transmission queue, if an AMS policy is defined for the transmission queue, any AMS protection is removed from the message prior to sending the message across the channel.
If the AMS policy is not defined for the transmission queue, the message is sent unchanged.

This value is valid for channels of type SDR or SVR.

ASPOLICY
Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on the target queue.
When the MCA receives an inbound message, if an AMS policy is defined for the target queue, AMS protection is applied to the message prior to the message being put to the target queue. If an AMS policy is not defined for the target queue, the message is put to the target queue unchanged.

This value is valid for channels of type RCVR or RQSTR.

User ID for AMS Interception on z/OS

The requirement for user IDs used with AMS Interception on z/OS are the same as those for existing AMS enabled applications. For a running channel, the sending MCA gets messages from a transmission queue and the receiving MCA puts message to target queues. The MCA user ID (MCAUSER) field, set on server-to-server channels, defines the user ID under which MCAs perform put and get requests.

With AMS Interception, AMS functions are performed during get and put requests, as with other AMS enabled applications. Therefore MCA user IDs have the same requirements as those for AMS application user IDs.

The MCAUSER used to perform the put and get is configurable, and dependent on whether it is an inbound or outbound channel. See the MQ Knowledge Center section on MCAUSER

for details on how the chosen user ID performs actions on the MCA. As such, the user ID that the channel initiator is running under, is the user ID that is to be used for AMS functions performed during server-to-server message channel interception.

Message size and MAXMSGL

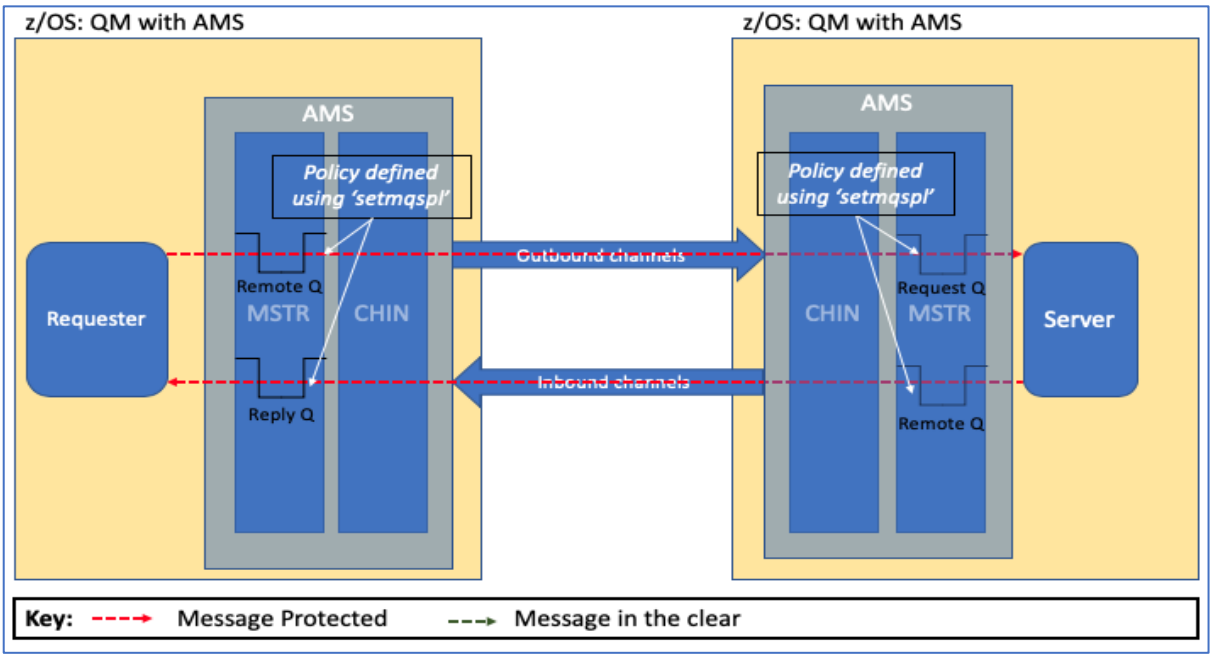
Due to AMS protection, the message size of protected messages will be larger than the original size.

Protected messages are larger than unprotected messages. Therefore, the value of the MAXMSGL attribute on both queues and channels, might need to be altered to take into account the size of protected messages.

The MQ for z/OS version 8.0 performance report (MP1J) offers some guidance as to the increase in size of protected messages in the “Does defining a policy on a queue affect the size of the message on that queue?”.

Example configuration – both queue managers are AMS-enabled

This initial example shows the configuration where both queue managers are AMS-enabled and a request/reply flow is used.



This configuration can be regarded as a single data island with end-to-end AMS protection – where the data is protected across the Enterprise.

Note that in this example, there are 2 queues defined on each queue manager with AMS policies – the remote queue for outbound messages and the inbound queue for messages arriving from the partner.

This example has the request message protected from the time it is put by the requester application until it is successfully gotten by the server application and similarly when the reply message is sent back from server to requester.

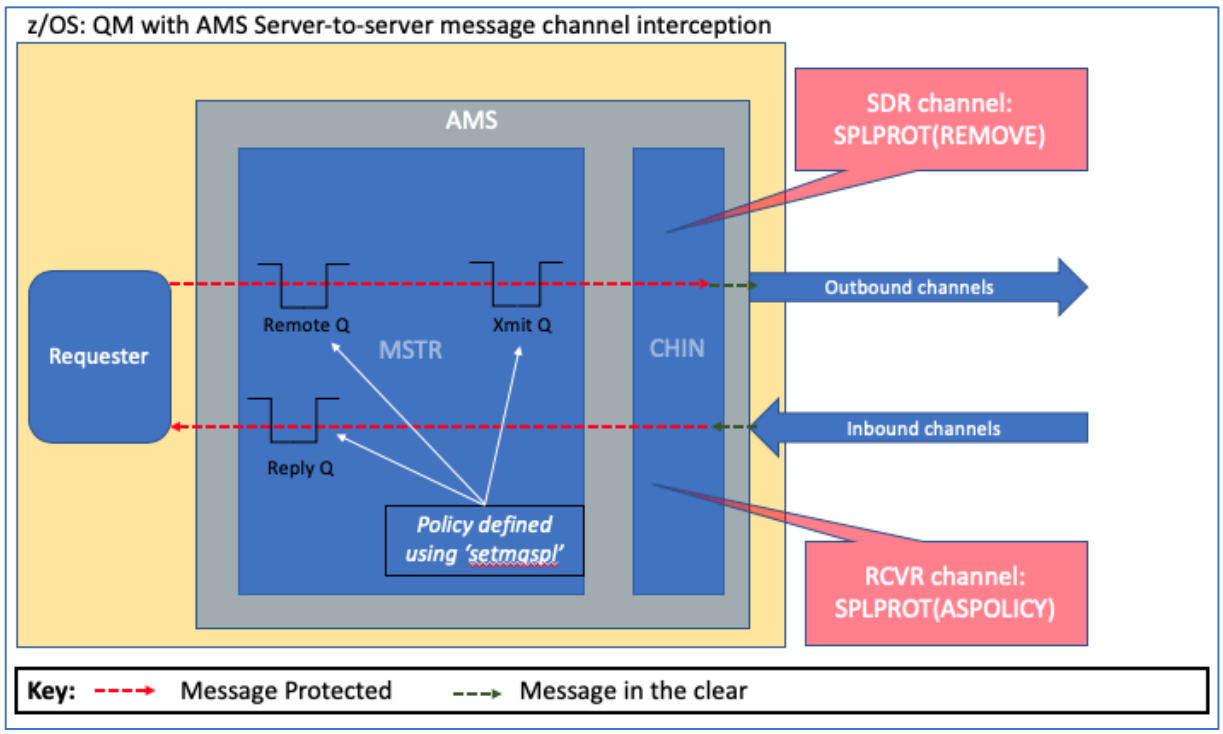
Example configuration – only one queue manager is AMS-enabled

Contrast the previous full AMS protection configuration with one using AMS Interception on z/OS, where only one queue manager is AMS-enabled.

The following diagram provides an example for AMS Interception on z/OS where the remote queue manager (not shown) is non-AMS enabled.

In this example, note that there are **3** queues that have AMS policies defined – on the outbound messages, the remote queue and the transmission queue both have policies defined, as does the reply queue for inbound messages.

Since the remote queue manager is non-AMS enabled, outbound messages need to have their AMS protection removed before sending, so the sender channel has SPLPROT "REMOVE". For inbound messages, AMS message protection is required, so the receiver channel uses SPLPROT "ASPOLICY" to ensure the queue manager applies the AMS policy, if one exists.



Note that in addition to the both the SDR and RCVR channels requiring the SPLPROT attributes being defined, the transmission queue also has a policy defined, in order to support AMS server-to-server message channel interception.

This configuration is an example of multiple data islands, where one island is protected and the other island is not protected by AMS qualities of protection.

Example configuration – queue managers are independently configured with AMS Interception on z/OS

This is an example of the “multiple data islands with independently configured AMS Interception on z/OS” configuration.

This configuration is similar to the previous configuration in that the local queue manager is configured with AMS Interception on z/OS.

In addition, the remote z/OS queue manager is also configured with AMS Interception on z/OS, i.e.:

1. 3 queues with AMS policies defined:
    a. inbound queue,
    b. remote queue,
    c. transmission queue
2. The receiver channel will have SPLPROT(ASPOLICY).
3. The sender channel has SPLPROT(REMOVE).

AMS Interception on z/OS and the MQ channel initiator

From an MQ performance perspective, the largest change introduced with AMS Interception on z/OS is that the channel initiator is potentially performing the removal of AMS protection from outbound messages and the applying of AMS protection to inbound messages.

This AMS protection, whether being removed or applied, is performed by one of the adapter tasks in the channel initiator, and is selected from the next available task from the set of adapter TCBs as specified by the CHIADAPS parameter.

Whilst applying or removing AMS protection from the current message, the adapter task will be unavailable for other work, either using CPU or waiting for cryptographic work to be completed by the available cryptographic hardware.

MQ statistics trace class(4) data can be used to see the impact of AMS Interception on z/OS on the channel initiators' adapter tasks, for example:

Example: Adapter usage when applying AMS protection to inbound messages

Configuration	Average CPU time	Average Elapsed time	Comments
Unprotected	9	9	
Confidentiality with key reuse 32.	18	18	1. CPU time is higher than unprotected, but no waiting for offload to cryptographic hardware. The additional cost is due to encrypting the data.
Privacy	20	347	1. Increased time difference between CPU and Elapsed, due to waiting for cryptographic work or *local lock.* 2. Increased cost from signing the message and encrypting the key.

The impact of AMS Interception on z/OS on the adapter tasks is discussed further in "AMS Interception on z/OS, channel initiator and local lock".

How the AMS policy type and the increased load on the adapters can affect workloads is discussed further in the "Business Partner streaming to AMS Interception-protected Enterprise" section.

As a result of the increased load on the adapter tasks when using AMS Interception on z/OS, it may be necessary to ensure additional channel initiator adapter tasks are made available to avoid impacting other non-AMS specific workload.

Typically the simplest way to determine if there are sufficient adapter tasks available is when reviewing the adapter report and ensuring that at least 1 adapter task is unused for the SMF interval.

Note: More adapter tasks may be specified using the “ALTER QMGR CHIADAPS(integer)” command, however these tasks will not be available until the channel initiator address space is restarted.

What is the performance impact of AMS Interception on z/OS?

When implementing AMS Interception on z/OS, you will have experience of at least 1 of the following 2 configurations:

- No AMS protection
- End to end AMS protection

Enabling AMS Interception for z/OS from either configuration will result in some impact to your system, whether additional CPU cost, increased latency on the transaction or re-distribution of cryptographic workload.

Moving from no AMS protection to AMS Interception on z/OS

When moving from an environment with non-AMS enabled queue managers to AMS enabled queue managers, there will be a performance impact.

It is recommended to read the MQ for z/OS v9.1.0 performance report, which discusses both general guidance on the impact of AMS as well as the performance improvements implemented in version 9.1.

Moving from end to end AMS protection to AMS Interception on z/OS

When moving from an environment where AMS protection spans multiple queue managers to one where AMS Interception on z/OS allows messages to flow to non-AMS enabled queue managers, there is impact on both the queue manager implementing AMS message channel interception and the hosting LPAR.

Consider an environment where there is data flowing between 2 AMS enabled queue managers, where protection is applied on LPAR A and removed on LPAR B.

- On each LPAR, there will be AMS-related cost on the applications using the AMS protected queues, as well as both the AMSM and MSTR address spaces.

Compare this to the AMS Interception on z/OS configuration where the protection is both applied on LPAR A and then the channel initiator removes the AMS protection prior to sending to LPAR B.

- The original cost of protection will still be incurred by the putting application as well as the MSTR and AMSM address spaces.
- In addition the cost of unprotecting the message will be incurred by the channel initiator, as well as some in the MSTR and AMSM address spaces.

The net cost of protecting and unprotecting the message will be similar whether performed in the end-to-end configuration or in the AMS Interception on z/OS configuration.

Despite the net cost of AMS protection remaining similar, consider the impact to both the CPU and Cryptographic processors on the LPAR with AMS message channel interception.

- Since the unprotection is occurring on the same system as the protection, there will be additional CPU utilization.
- Similarly, load on the cryptographic processors may double. This additional load may add latency due to the requests being queued waiting for cryptographic processors.

Channel Throughput

As mentioned previously the work performed in the channel initiator relating to AMS Interception on z/OS is completed by adapter tasks.

Messages flowing over a single channel are processed in series, i.e. the current message must be processed in its entirety, whether getting from a queue and unprotecting, or protecting and putting to a queue, before the subsequent message is begun to be processed.

The increased time spent protecting or unprotecting the message by the adapter task means that the queued messages are processed more slowly.

If those queued messages are on a remote system, possibly belonging to a business partner, they may see their MQ queues deeper than before AMS Interception on z/OS was implemented.

There is potential for the queue on the remote system to fill and it is suggested that the owner of the remote system is made aware that AMS Interception on z/OS is being implemented and to monitor for deep queues.

It may be necessary to increase the maximum depth of the queue on the remote system or to modify the application to wait-then-retry in the event of queue full.

If message expiry is used on the remote system, it may be necessary to review the value used.

AMS Interception on z/OS, Channel initiator tasks and local lock

Typical guidance for tuning adapter tasks is to verify that at peak volumes there is at least 1 unused adapter task. This can be verified using the statistics class(4) data.

Many z/OS subsystems including the channel initiator uses local lock to serialize access to certain resources, such as storage allocation.

This local lock may be taken by the adapter tasks, and with the additional work performed by the AMS Interception on z/OS, increased local lock time can result.

Therefore, the configuration of the channel initiator tasks such as the adapters (CHIADAPS) and dispatchers (CHIDISPS) when using significant numbers of channels using the SPLPROT attribute can affect the amount of local lock time, particularly if additional channel initiator tasks are added.

The tables and charts that following in this section aim to demonstrate the impact of local lock times on the adapter tasks when running a simple request/reply workload across 50 channel pairs, protected by AMS Interception on z/OS, when using AMS Confidentiality with key reuse (KR) of 32. This configuration should mean a relatively light usage of any Crypto Express hardware.

A symptom of increased local lock time can be seen in the class(4) statistics data with a significant difference between the adapter CPU time and the adapter elapsed time.

Why might the adapter elapsed time be significantly larger than the CPU time?

When there is sufficient CPU, the statistics class(4) adapter report will generally show the average elapsed time to be similar to the average CPU time, but there are a number of times where there will be a more significant difference between the 2 values. Some of these reasons include:

1. Persistent messages, waiting for log writes
2. Message selection – this is discussed in performance report MP16.
3. Buffer pools used by the queue are at capacity and the queue manager is enforcing immediate writes to page set.
4. AMS Interception for z/OS – waiting for local lock and/or cryptographic work offloaded to Crypto Express hardware.

How do we know this is the impact from local lock?

Local lock can be relatively difficult to quantify in terms of the impact to the messaging rate, but with regards to these measurements we know that the increased transaction round trip time occurs despite:

- Cryptographic hardware not constrained,
- CPU is lightly loaded and for less than 15% of the interval are there more tasks active than CPUs available.
- Workload is not performing disk I/O,

- Dedicated performance network is capable of significantly more throughput.

We can confirm the presence of local lock by formatting a standalone channel initiator dump using IPCS “SYSTRACE PERFDATA” function and checking the time spent in local lock.

Varying the number of dispatchers.

In the following table we demonstrate how varying the number of dispatchers can affect the usage of the adapter tasks and the overall throughput rate.

All of the measurements in the following table and chart are configured with 10 adapter tasks.

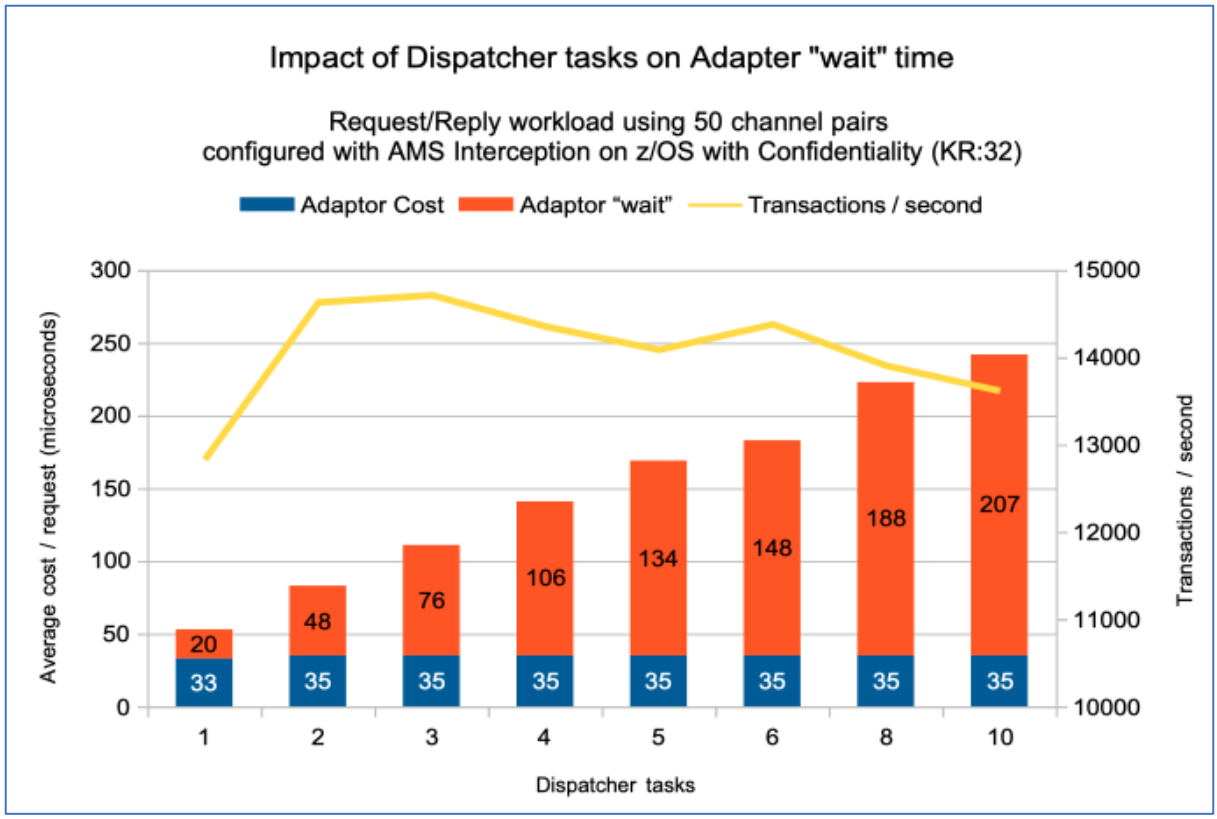
Dispatcher tasks	Transactions per second	Cost per transaction microseconds	Adapter Average Elapsed microseconds	Adapter Average % Busy
1	12839	454	53	20
2	14639	469	83	37
3	14721	469	111	49
4	14362	479	141	58
5	14095	477	169	72
6	14387	473	183	80
8	13913	477	223	93
10	13623	471	242	100

In the above table, the average CPU time per adapter request remains constant at 35 microseconds, but as the number of dispatcher tasks increases, the average elapsed time per adapter request increases significantly, from 53 to 242 microseconds.

The average usage of the 10 available adapters also increases significantly – note that with 3 dispatchers the peak throughput is achieved but the 10 adapters are only 49% busy. When the number of dispatchers is increased to 10, the overall throughput has dropped but the 10 adapters are all 100% busy – and spending approximately 200 microseconds per adapter call waiting for the local lock.

In none of these measurements are the configured dispatcher tasks more than 62% utilized over the SMF intervals.

The following chart is a representation of the key items from the table whilst varying the number of dispatchers.



Notes on chart:
Wait time is calculated by subtracting the average CPU time from the average elapsed time in the statistics class(4) adapter report.

The chart shows that the peak rate was achieved with 2-3 dispatcher tasks – and this was despite the adapter tasks being less than 50% utilised. These configurations saw relatively small adapter wait (local lock) times. As more dispatchers were made available to drive the adapter tasks, the time spent waiting for the lock increased significantly and the adapter usage increased – solely in wait time.

With an adapter task spending more time waiting for the local lock, the adapter is unable to process work from other channels which may not need to take the local lock, thus potentially affecting other channels that are not using the AMS Interception on z/OS function.

Varying the number of adapters.

In this table, the number of dispatchers is set to 3 but we demonstrate how varying the number of adapters can also affect the usage of those available adapter tasks, and to a certain extent the overall throughput rate.

Adapter tasks	Transactions per second	Cost per transaction microseconds	Adapter Average Elapsed microseconds	Adapter Average % Busy
1	10022	439	33	100
2	13291	454	50	100
3	14000	465	71	100
4	14215	465	93	100
5	14585	469	102	90
6	14487	474	108	79
8	14746	464	110	61
10	14527	474	111	49

In the above table, the dispatcher tasks were never more than 25% busy for the measurements. The average CPU time per adapter request was typically 34-35 microseconds[1], but by limiting the number of available dispatcher tasks, we have reduced the effect of local lock on the adapter tasks as a side effect of constraining the number of concurrent channels flowing through the channel initiator.

[1] Except in the single adapter configuration where the average cost was 29 microsecond – but this comes at significantly less throughput.

Guidance for adapters and dispatchers with AMS Message Channel Interception

It is worth noting that the local lock does not necessarily cost in terms of CPU – it is primarily time waiting for resource, but does block the adapter task from performing other work.

Guidance for configuring the number of adapter and dispatcher tasks is available in the "MP16 – the MQ Capacity planning and tuning guide", but a reminder is provided below:

Dispatchers:

- As few as possible, as long as the available dispatchers are not used at capacity.

As the earlier charts have shown, too few dispatchers can inhibit throughput.

It may be that limiting the number of dispatchers can be used to reduce local lock time, or at least the "wait" time in the adapter tasks that we have discussed, but this can affect non-AMS Interception on z/OS channel throughput.

Adapters:

General guidance for adapters depends on the type of workload.

For example, workload using persistent messages can take much longer than those for non-persistent messages, due to log I/O, thus a channel initiator processing a large number of persistent messages across many channels may need more than the default 8 adapter tasks for optimum performance.

Similarly, workloads using AMS Interception on z/OS to apply or remove AMS protection to messages may benefit from additional adapter tasks.

Where serialization for resource occurs in the channel initiator, such as with AMS Interception on z/OS or persistent messaging and as per MP16 guidelines, we suggest CHIADAPS(30) for systems with up to 4 processors and then increase CHIADAPS by 8 for each additional processor up to a maximum of CHIADAPS(120). We have seen no significant disadvantage in having CHIADAPS(120) where this is more adapter tasks than necessary.

At a minimum, we suggest ensuring that there is **at least 1** adapter task available the time of peak workload.

By ensuring there are unused adapter tasks even at peak volumes, this should prevent adapters serialized and waiting for resource (local lock) whilst processing AMS Interception on z/OS workload from impacting workload on channels that are not AMS Interception on z/OS-enabled.

Finally, and to re-iterate, we would advise monitoring the utilization of the channel initiator tasks using the statistics class(4) data to determine whether there are sufficient tasks available. If the tasks are constrained and there is sufficient CPU available, some benefit may be achieved with small incremental changes to the number of available channel initiator tasks.

Indexing of SYSTEM.PROTECTION.POLICY.QUEUE

When starting your queue manager, you may see message:

```
CSQI004I @VTS1 CSQIMGE3 Consider indexing SYSTEM.PROTECTION.POLICY.QUEUE by
               CORRELID for BATCH connection VTS1AMSM, xxx messages skipped
```

The data held on this queue is also cached within the AMSM region, so the majority of access uses cached data.

The queue is only read by the AMSM address space on start-up and when the REFRESH command is requested.

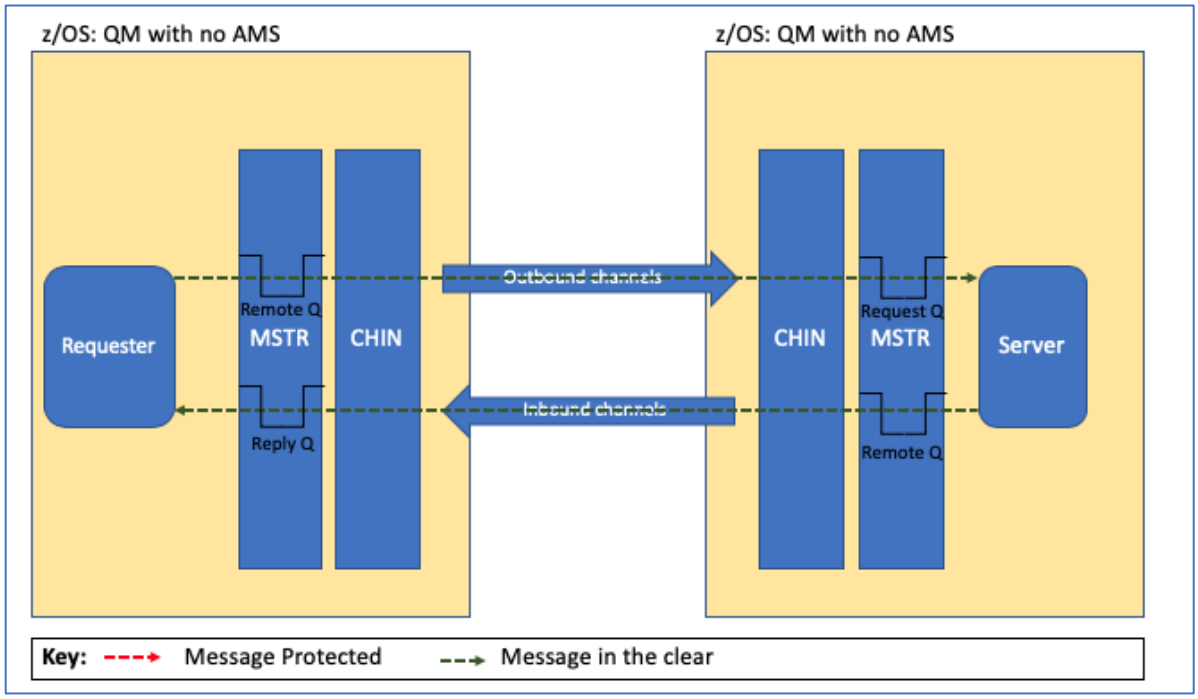
As a result, unless there are hundreds of policies defined, it is unlikely that indexing the SYSTEM.PROTECTION.POLICY.QUEUE will offer an benefit.

Example scenarios using AMS Interception on z/OS

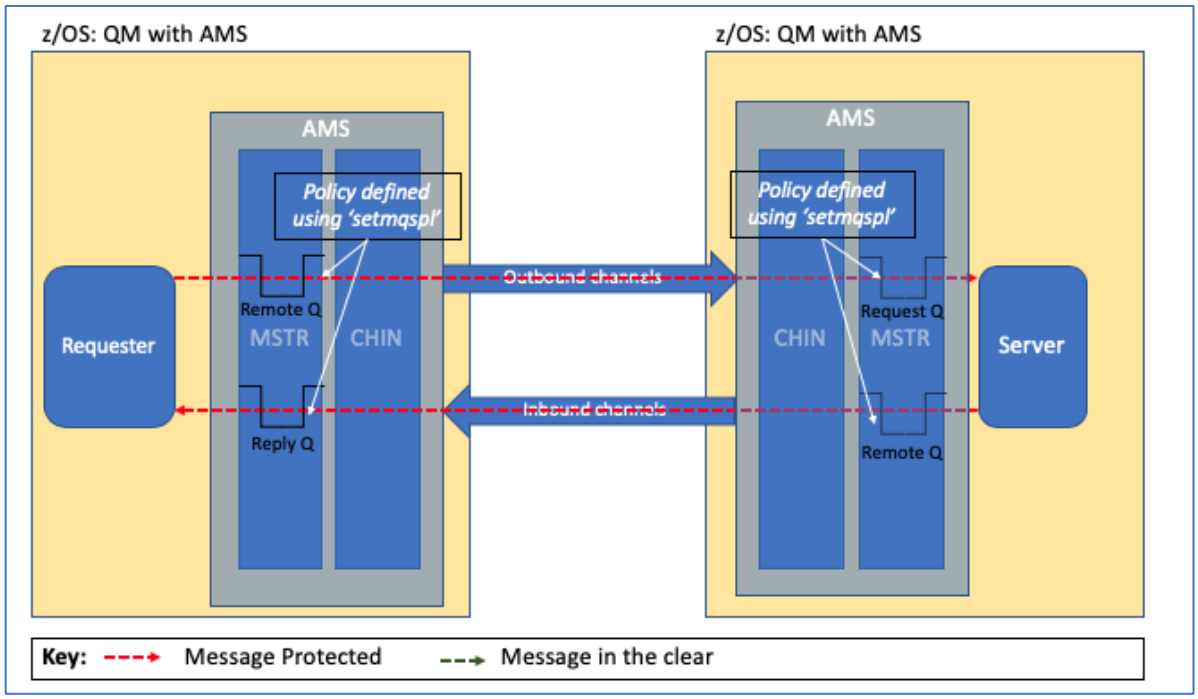
Single AMS protected data island - Request / Reply workloads

This scenario uses a simple request/reply workload to demonstrate the impact of AMS Interception on server to server message channels when compared with a non-AMS workload and a full end-to-end AMS configuration.

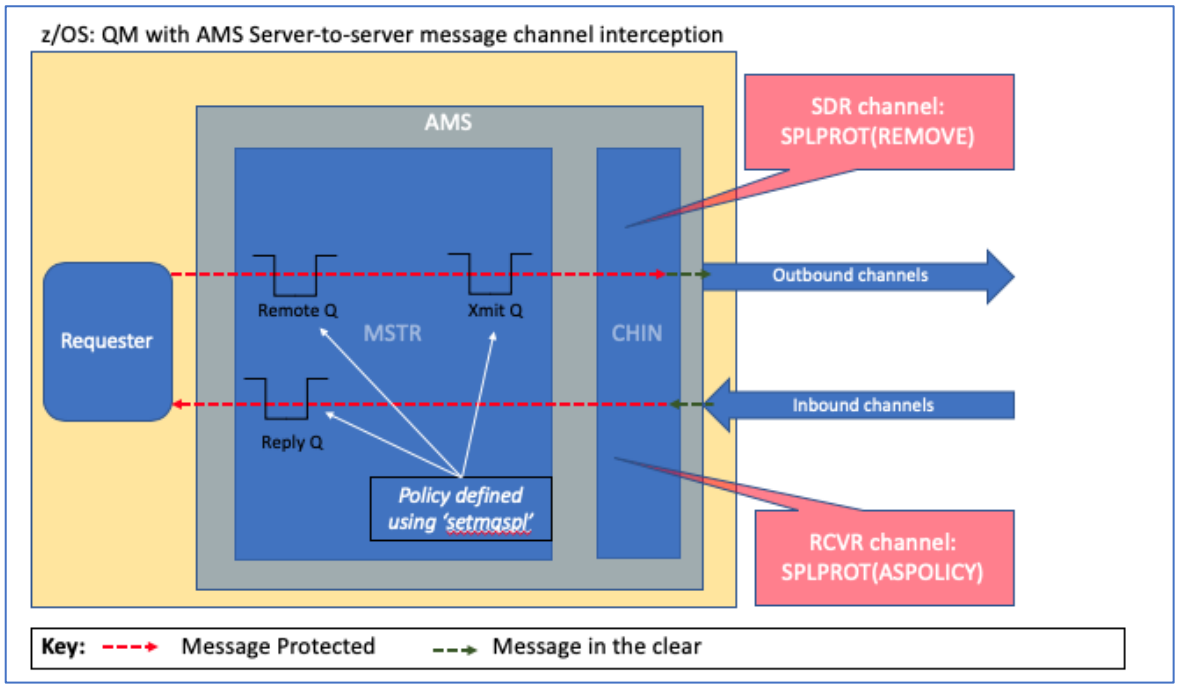
The initial workload (*single data island, with unprotected workload),* configured without AMS protection can be represented thus:



The full end-to-end AMS (*single data island, with end-to-end AMS protection)* configuration can be represented:



When using AMS Interception on z/OS on the requester-side of the configuration, the configuration is represented, with the server side having no AMS protection configured (*Multiple data islands, one protected with AMS Interception on z/OS):*



The measurements that follow use a single pair of request / reply tasks with a 32KB non-persistent message per channel pair.

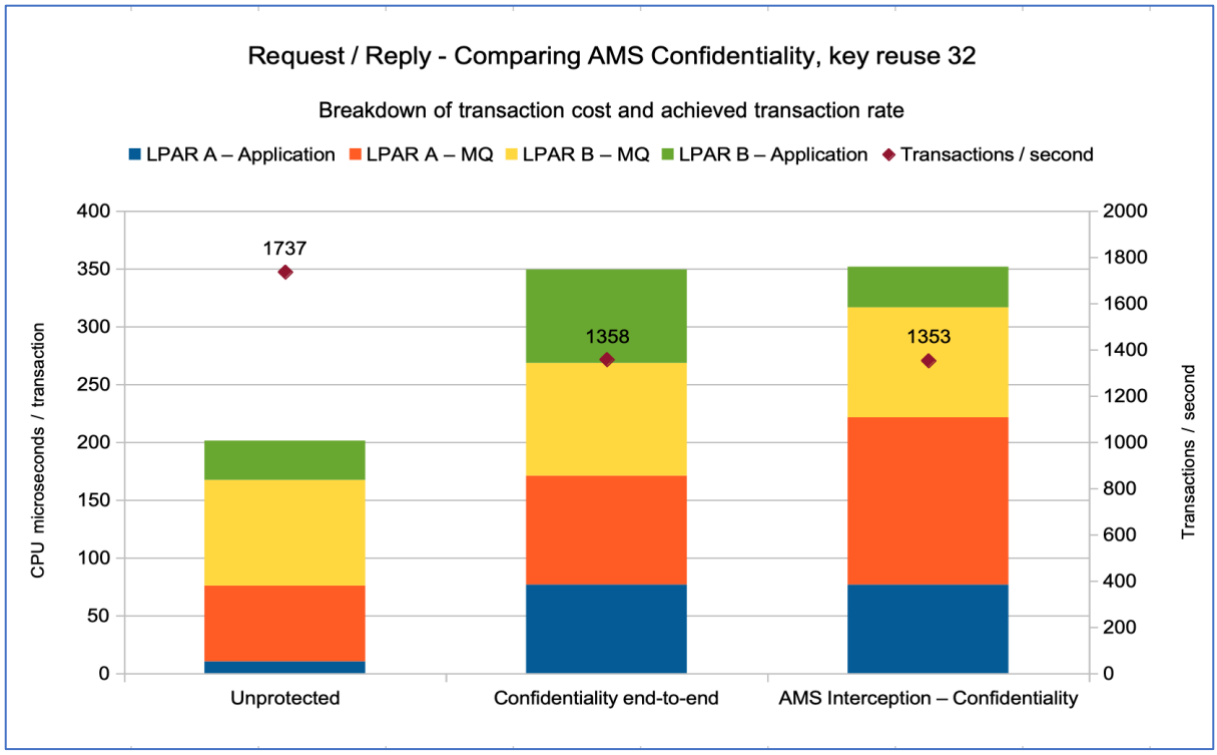
For the request / reply workloads we focus primarily on the AMS Confidentiality quality of protection as these are the most likely use cases.

AMS Confidentiality with key reuse 32

For the purposes of the following measurements, a key reuse of 32 has been used, which benefits from a reduced use of asymmetric cryptography whilst still re-generating the key on a regular basis.

As will be demonstrated later, the performance of AMS Confidentiality with key reuse of 32 is significantly better than AMS Privacy, regardless of whether the AMS protection is end-to-end or on a single end via the AMS Interception on z/OS.

As such, the impact from AMS Confidentiality over the unprotected system is much smaller, and is demonstrated in the following chart:



Notes on chart:
MQ costs relate to those in the MSTR, CHIN and AMSM address spaces.

Applying AMS Confidentiality with key reuse of 32 to the workload, saw the transaction cost increase by 150 CPU microseconds, which for this workload equates to a 75% increase.

It should be emphasized that the increase of 150 CPU microseconds would be seen with the 32KB non-persistent workload whether the application cost was 10 microseconds or 1000 microseconds.

It should also be noted that this additional 150 CPU microseconds is for the request/reply workload which involves protecting and unprotecting the message twice – once on the outbound message and again on the inbound message. It may be assumed that a single direction flow would cost of the order of 75 CPU microseconds.

This increased cost resulted in the transaction rate dropping by 22% when compared to the unprotected measurement.

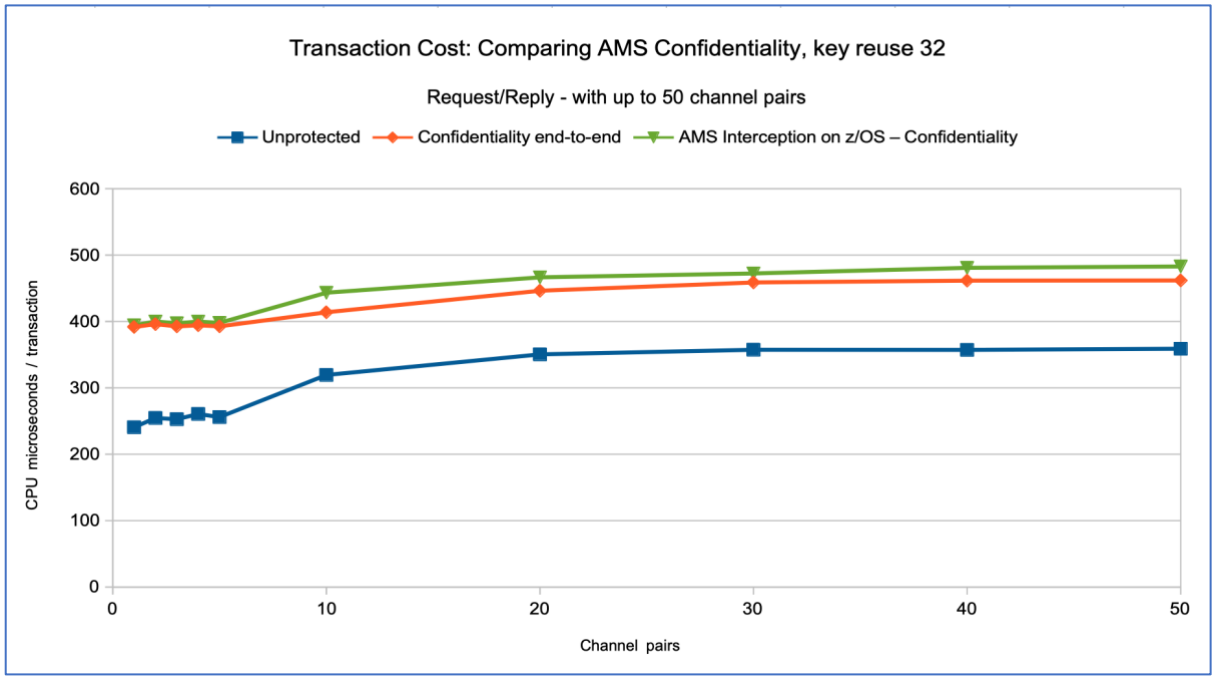
Does AMS Confidentiality scale similarly to the "no protection model"?

With the increased load in the channel initiator for AMS Interception on z/OS, the throughput rate when running under significant load can be affected for a number of reasons including:

1. Increased load on the channel initiator adapter(s) from applying / removing the encryption from the message.
2. Increasing the number of adapter tasks can result in increased local lock time.

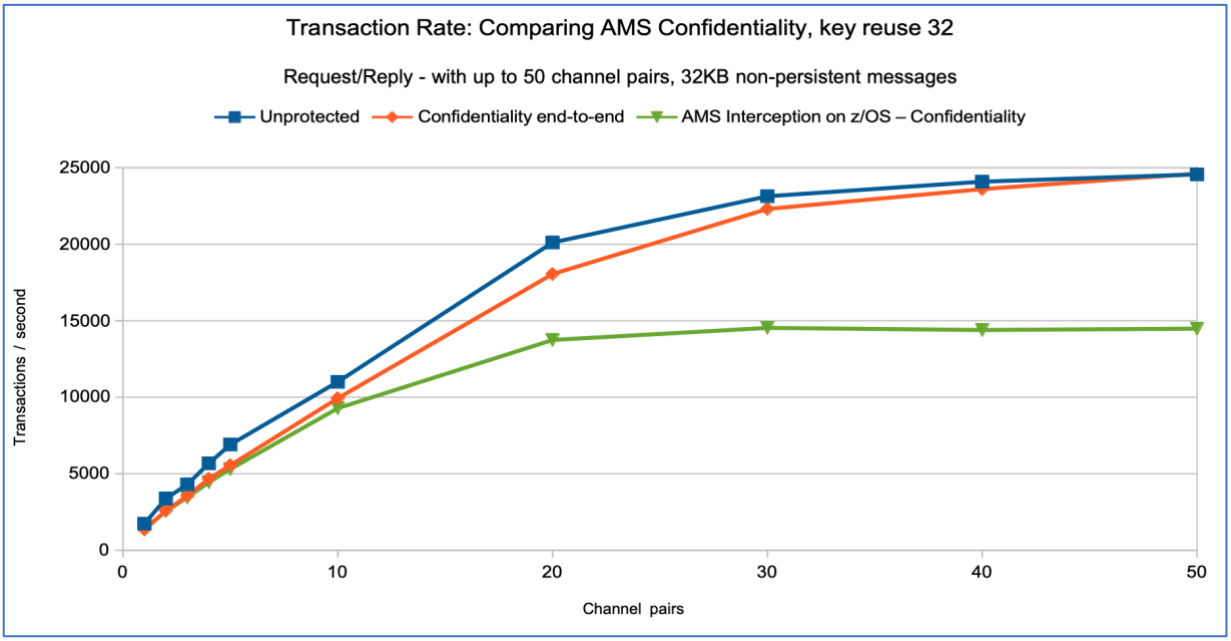
The first chart shows that the transaction cost for the AMS Interception on z/OS with AMS Confidentiality measurement is similar to that the AMS Confidentiality end-to-end measurement.

Transaction cost:



However the second chart showing the achieved transaction rate shows markedly worse performance when running with significant numbers of channels where AMS Interception on z/OS is applied.

Transaction rate:



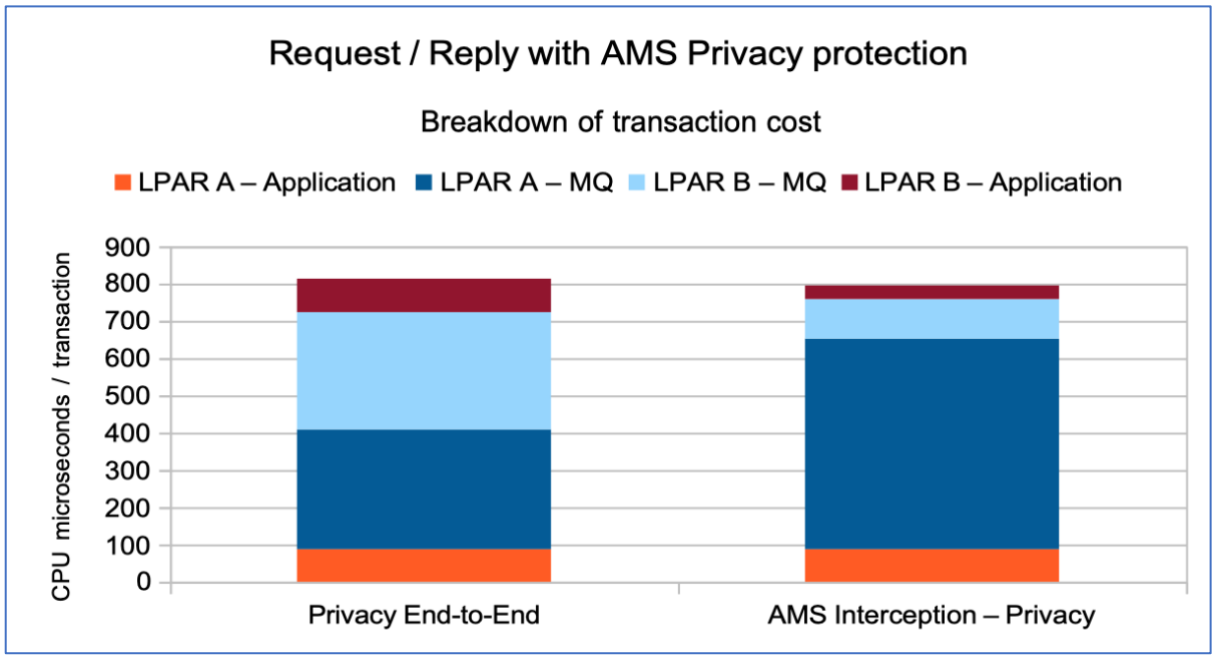
Notes on transaction rate chart:

- In this instance the reason for the AMS Interception on z/OS measurement flattening out at 15,000 transactions per second is due to increased local lock in the channel initiator.
- We found that reducing the number of adapter tasks, from 20 to 5, did reduce the local lock but did not improve the throughput.

AMS Privacy

As mentioned previously, moving from an AMS end-to-end protection model to an AMS Interception on z/OS model where the AMS protection is only on one queue manager, the net cost impact from AMS remains similar.

The following chart demonstrates the balancing of the cost between the AMS end-to-end and AMS Interception on z/OS models.

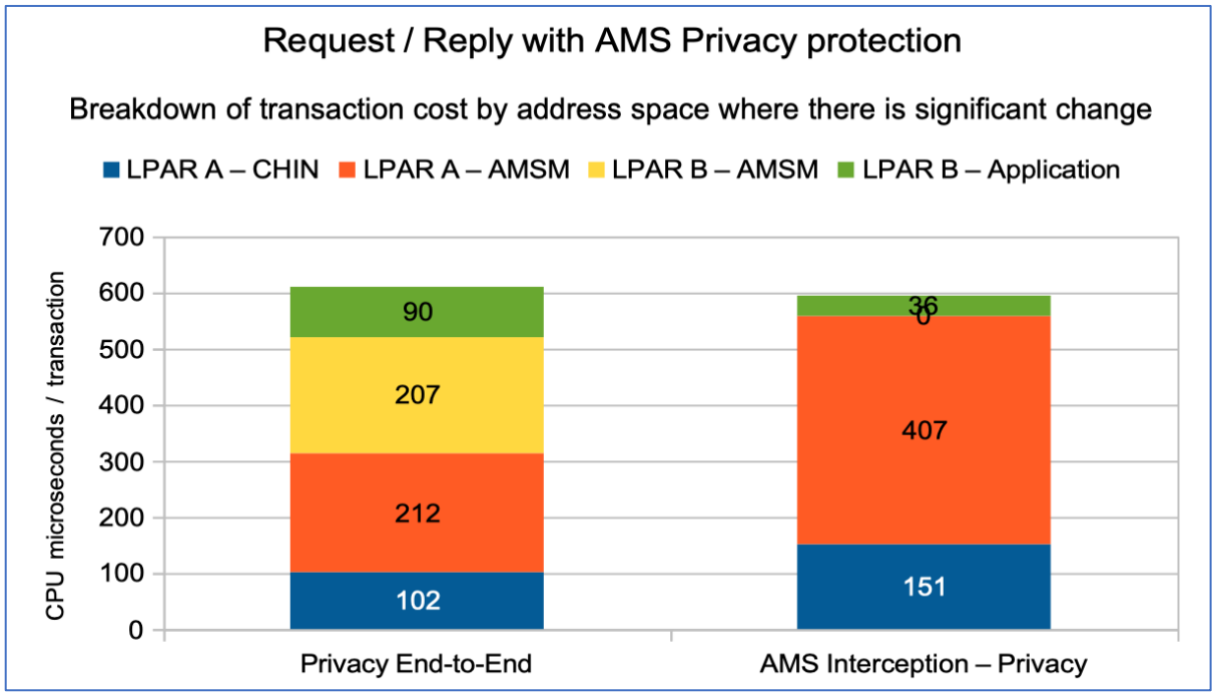


Note on chart:
The MQ cost include the costs of the MQ MSTR, CHIN and AMSM address spaces.

The total transaction cost remains similar whether the AMS protection is applied across both LPARs or only on LPAR A.

Moving from the end-to-end protection model to the AMS Interception on z/OS model, we see cost move from LPAR B's MQ and application into LPAR A's MQ.

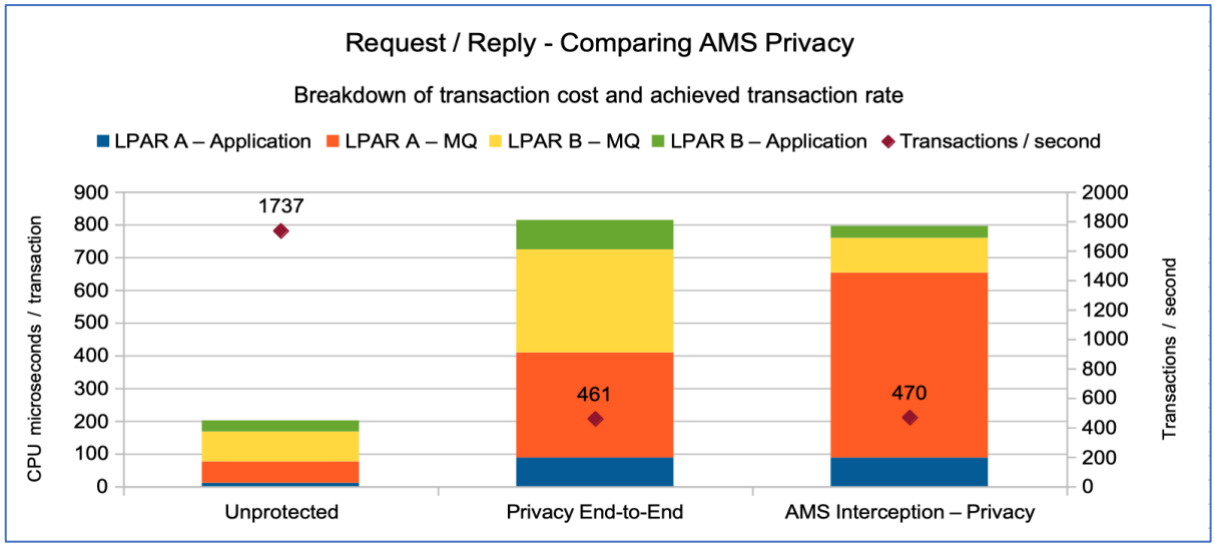
Using the data from the previous chart and expanding to show only address spaces where there was *significant change* to the cost, results in the following chart:



Notes on chart:
This chart shows that there is increased cost in LPAR A's channel initiator address space when moving to the AMS Interception on z/OS model. This increased cost is offset by the reduced cost in LPAR B's application.

The net cost attributed to the AMS address space remains consistent – again the cost shifting from LPAR B to LPAR A for the AMS Interception on z/OS model.

At this point we have seen how moving from the AMS Privacy end-to-end protection model to the AMS Interception on z/OS using AMS Privacy model shifts the balance of the AMS cost from LPAR B to the protected LPAR A, but how does this compare to costs and the transaction rate when running with no AMS protection?

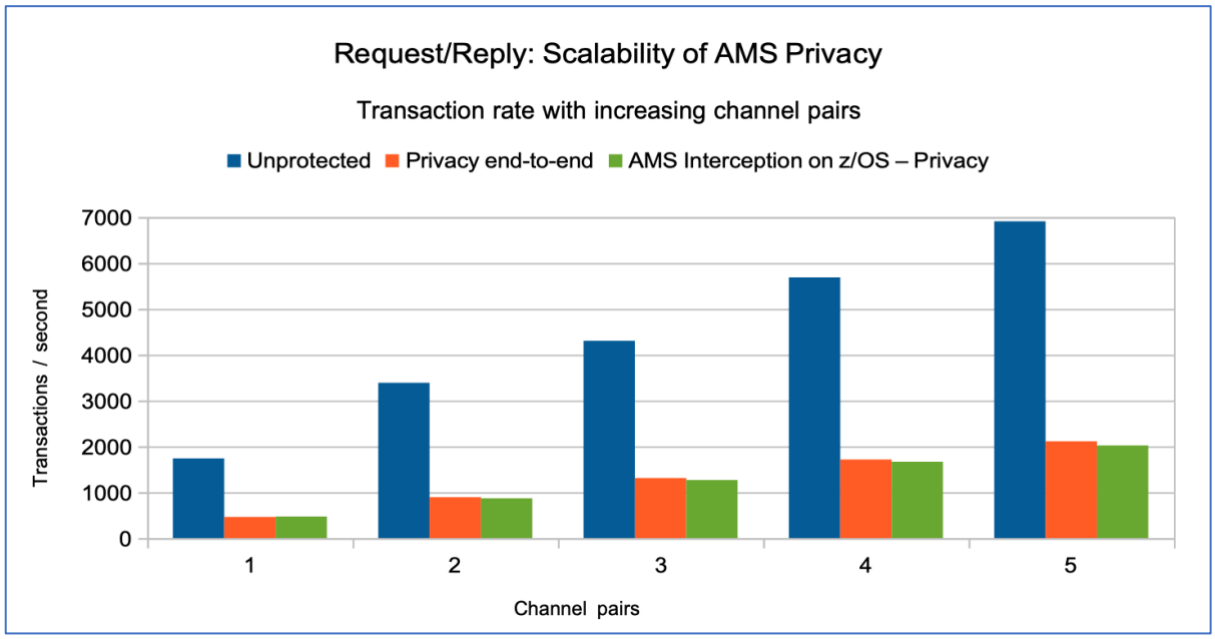


In these measurements, the impact of applying AMS Privacy to a simple request/reply workload is to increase the transaction cost 4 times and reduce the transaction rate to one quarter of the unprotected rate.

It should be noted that these measurements, the application cost appears to grow significantly when using AMS protection. However the *application* in use, has no processing outside of the message processing, and the increased cost of AMS protection would be similar (+75 microseconds for protect and unprotect) regardless of whether the application cost 10 microseconds or 1000 microseconds per “transaction”.

Does AMS Message Channel Interception scale similarly to the "no protection" model?

Provided there is sufficient resource, then the AMS Interception on z/OS model should show a similar pattern as workload is increased, subject to local lock waiting for serialized resources, for example:



Notes on chart:
The chart shows the achieved transaction rate as more request/reply workloads are added, each using their own set of MQ channels.

Whilst the unprotected configuration appears to scale at a faster rate, the achieved throughput with 5 channel pairs is 4 times that of 1 channel pair.

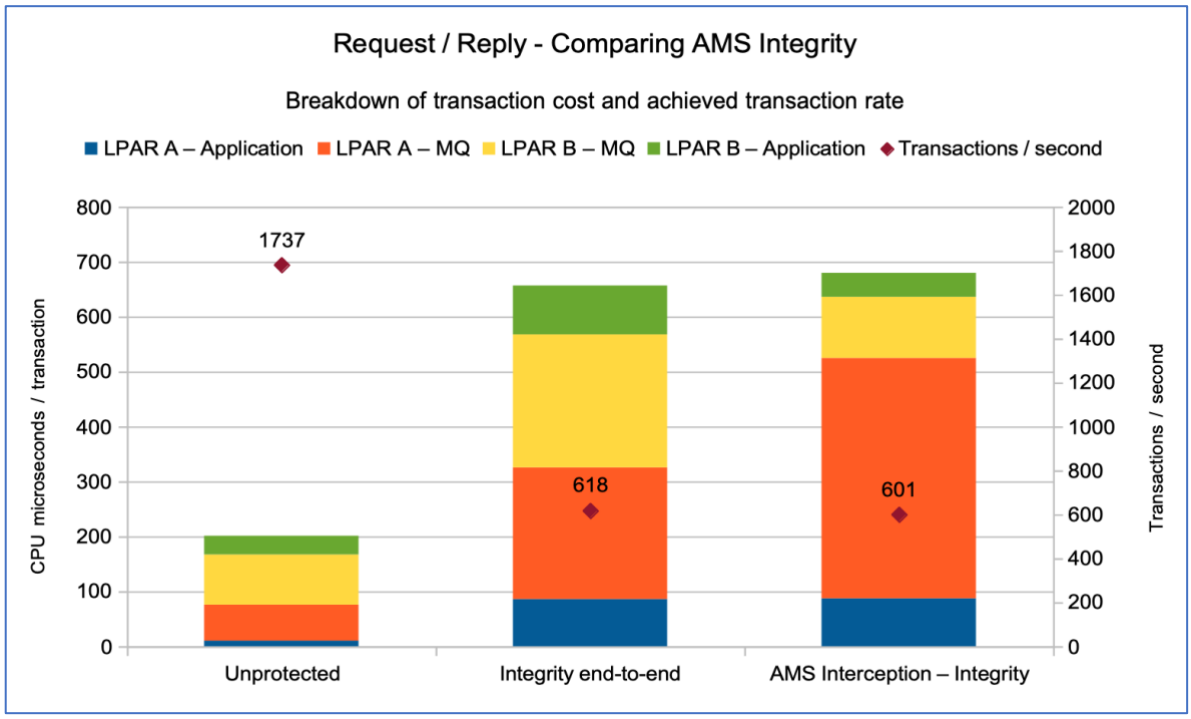
For the AMS Privacy end-to-end configuration, the achieved throughput with 5 channel pairs is 4.6 times that of 1 channel pair.

The AMS Interception on z/OS using privacy configuration achieved 4.3 times the throughput when moving from 1 to 5 channel pairs.

Given the transaction cost for the 5 channel pair measurements is the same for the AMS Privacy end-to-end configuration and the AMS Interception on z/OS with Privacy configuration, the reason for the difference in transaction rate, approximately 5%, is due to all of the cryptographic work being performed on the single LPAR – placing more load on the available Crypto Express coprocessors.

AMS Integrity

Moving from AMS Integrity end-to-end to AMS Interception on z/OS with Integrity protection shows the same net cost effect that we have seen with both privacy and confidentiality, as demonstrated in the following chart.



Notes on chart:
This chart shows that there is increased cost in LPAR A's channel initiator address space when moving to the AMS Interception on z/OS model. This increased cost is *largely* offset by the reduced cost in LPAR B's application.

The net cost attributed to the AMS address space remains consistent – again the cost shifting from LPAR B to LPAR A for the message channel interception model.

Transaction rate between the AMS Integrity end-to-end and the AMS Interception on z/OS with Integrity measurements is similar.

Two data islands, independently protected using AMS Interception on z/OS

This configuration may appear to be preferable when protecting data in an Enterprise as it can mean that certificate management does not have to be coordinated across data islands, nor do the data islands need to run the same quality of AMS protection.

There are a number of reasons why using independently protected data islands may not perform as well as an Enterprise-wide data island, including:
- Increased load on cryptographic hardware
- Increased MQ cost in MQ channel initiator(s) and AMSM regions.
- Increased latency on end to end message flows.

For example, consider the additional processing involved when comparing a two independently protected data islands over a single Enterprise-wide data island when sending a message between 2 queue managers:

- Additional unprotect in sending-side channel initiator prior to send over network.
- Additional re-protect in receiving-side channel initiator prior to MQPUT to target queue.

In each case, these additional pieces of work will add cost to the flow as well as increasing the time taken to flow the messages between applications.

Depending on the AMS quality of protection and in the case of AMS Confidentiality the key-reuse value, these additional unprotect and re-protects may result in the doubling of the use of the associated cryptographic processors when compared to the single Enterprise-wide data island use case.

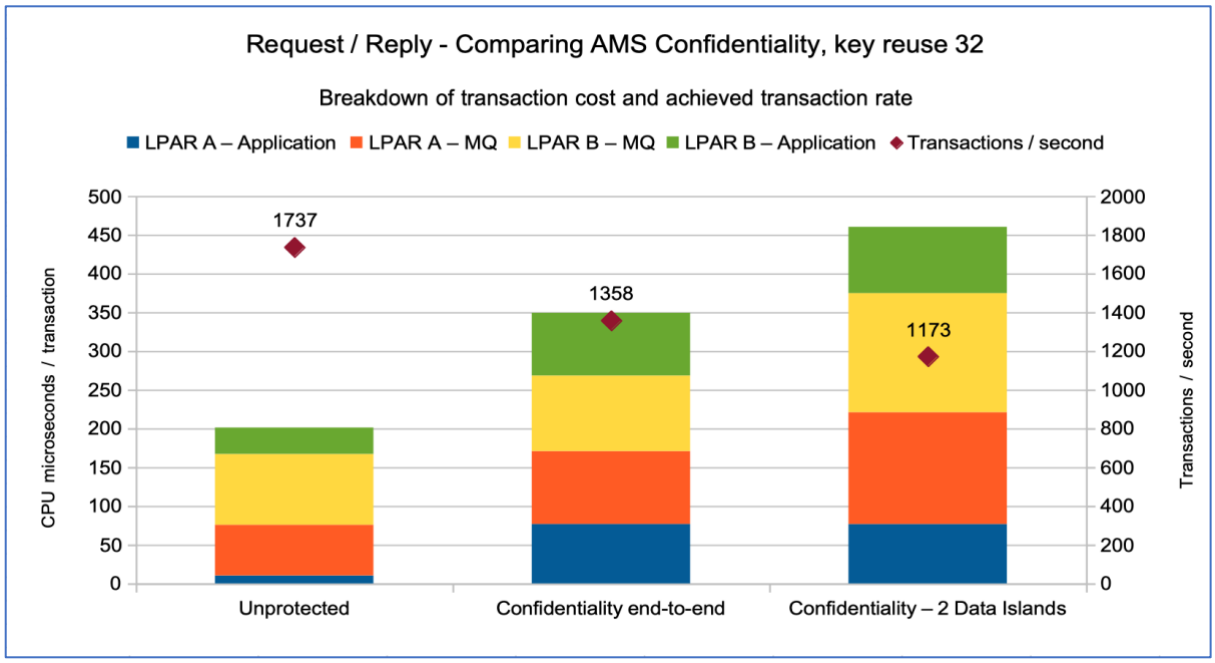
The measurements in this section compare the following 3 scenarios:
1. Unprotected single data island, request/reply workload
2. Enterprise-wide single data island, request/reply workload
3. Two data islands, independently protected using AMS interception on z/OS.

As with the earlier request/reply workloads, the initial measurements that follow use a single pair of request / reply tasks with a 32KB non-persistent message per channel pair using the AMS Confidentiality quality of protection and key reuse is set to 32 as this provides a good balance of protection at minimal cost.

Comparison of single channel pair performance

The use of two independently protected data islands, when compared with an end-to-end AMS protection model, adds both cost and latency to the transaction.

The following chart demonstrates the impact on the cost and the achieved transaction rate when implementing two independently protected data islands.



Notes on chart:
MQ costs are based on the average transaction cost in the MSTR, CHIN and AMSM address spaces.

By using multiple data islands, the transaction rate drops by 13% from the achieved rate when using end-to-end AMS protection.

With regards to transaction cost, there is additional cost in both LPARs for primarily the channel initiators, but also the AMSM regions. In this example this additional cost amounts to 110 CPU microseconds, or a 31% increase in transaction cost.

The transaction cost on LPAR A is, unsurprisingly similar to that observed in the earlier measurements when using 2 data islands where AMS Message Channel Interception was applied on LPAR A only.

The costs observed in LPAR B are similar to those costs in LPAR A.

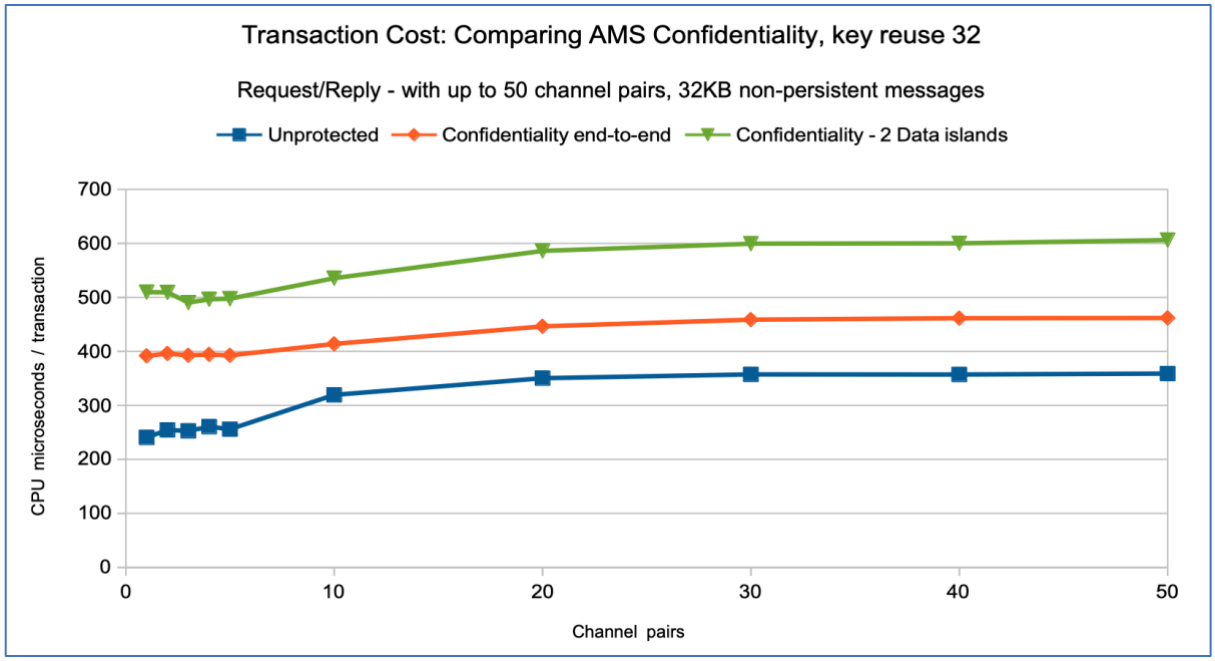
Scalability when using two independently protected data islands

Earlier measurements comparing the performance of multiple data islands, where only one was protected using AMS Interception on z/OS showed a significant difference in the performance when compared with end-to-end AMS protection using the same AMS quality of protection. This was in part due to the increased difference between the adapter CPU time and elapsed time, which was largely due to increased local lock.

With the increased load on **each** channel initiator in the independently AMS Interception on z/OS -protected data island, we see a larger impact on both the transaction cost and transaction rate when compared with AMS protection end-to-end.

The first chart shows that the transaction cost for the AMS Confidentiality 2 data islands measurement is markedly higher than the AMS Confidentiality end-to-end measurement.

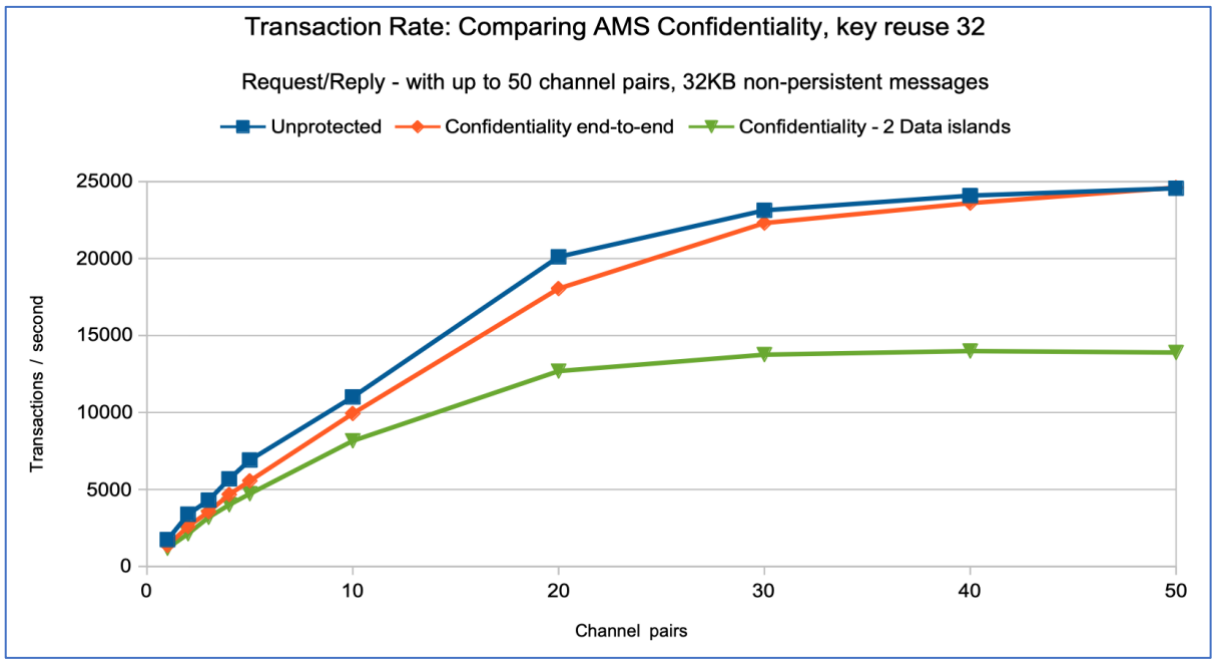
Transaction Cost:



Notes on chart:
For the two independently protected data islands, the increase in transaction cost is of the order of 100 to 140 microseconds per transaction over the equivalent configuration where the data is protected end-to-end in a single data island.

In the second chart, showing the transaction rate, we see a similar pattern to that in the "*Multiple data islands, one protected with AMS Interception on z/OS"* configuration but in terms of throughput we have seen a drop in overall throughput of up to 17% compared to the single AMS Interception on z/OS' throughput.

Transaction Rate:



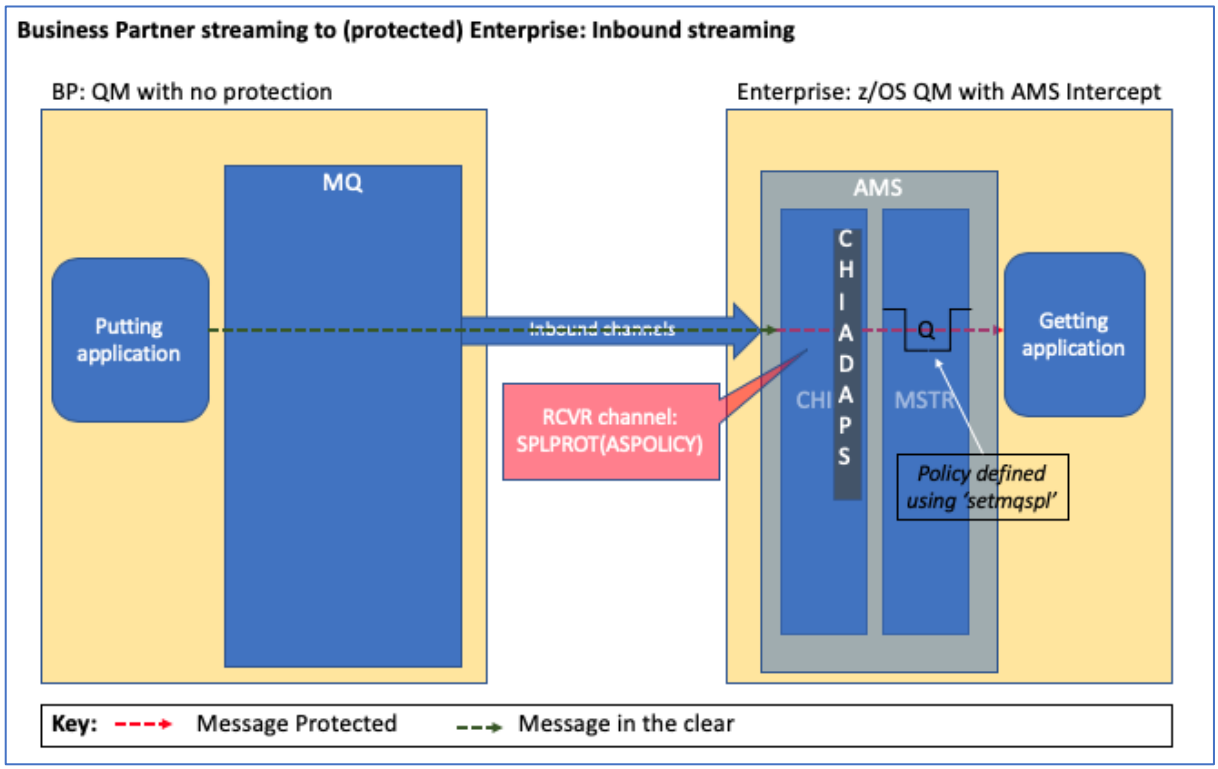
Notes on chart:
Once more we see the transaction rate for the AMS Interception on z/OS measurements tail off far earlier than in the end-to-end measurement.

In the two independently protected data islands measurements, both channel initiators see increased disparity between adapter CPU time and elapsed time, indicating impact from local lock.

Business Partner streaming to AMS Interception-protected Enterprise

This scenario aims to demonstrate the impact to a business partner streaming messages to an Enterprise-hosted queue manager configured with AMS Interception on z/OS.

The configuration used can be represented thus:



In these measurements, the business partner is running on a distributed non-AMS enabled queue manager.

The putting application on the distributed partner is attempting to put messages at a rate of 10,000 2KB non-persistent messages per second.

These messages flow over a SDR-RCVR type channel to the AMS-enabled queue manager, where the channel initiator applies AMS protection prior to the message being put on the target queue.

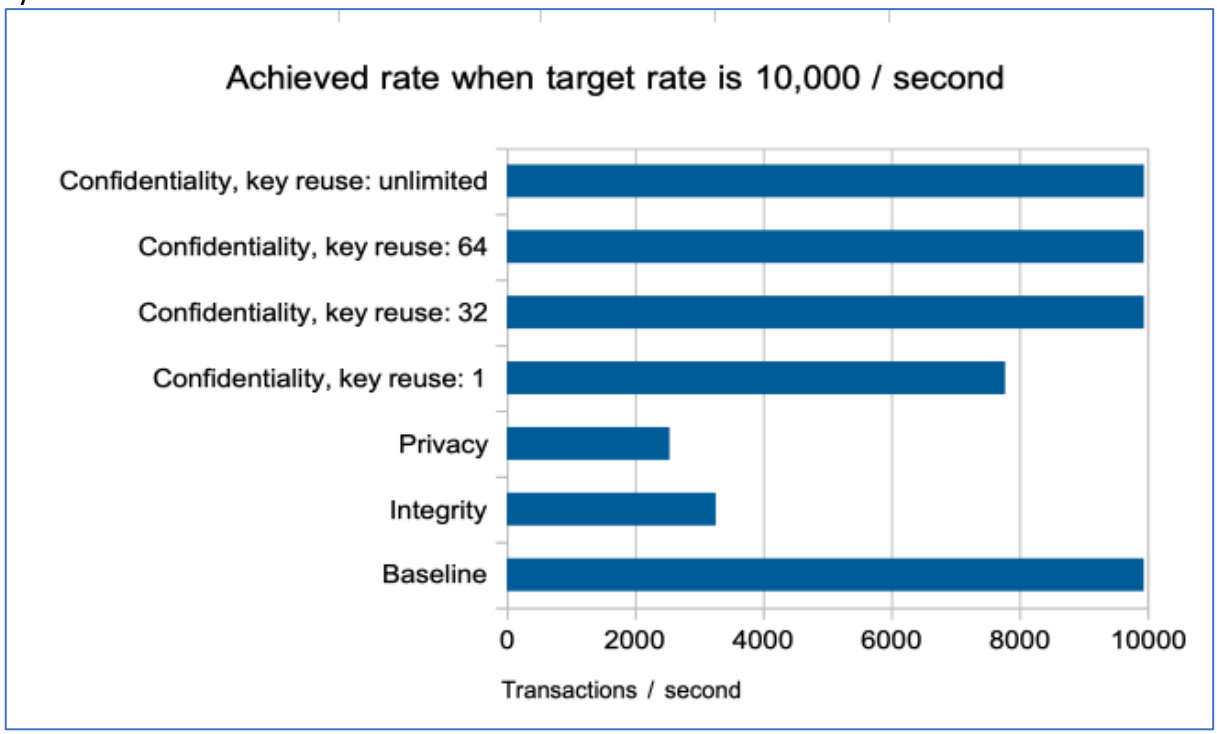
The authorized application gets and unprotects the message, discarding it after use.

The types of AMS protection used in these measurements are:

- None (baseline)
- Integrity, message is signed using SHA256
- Privacy, message signed using SHA256 and encrypted with AES256
- Confidential, message encrypted with AES256 and key reused:
  - 1 time
  - 32 times

- 64 times
- Unlimited.

The first chart in this section shows the achieved throughput rate on the AMS-enabled system.



This chart shows that the target rate of 10,000 messages per second was not sustainable in 3 of the 7 configurations, namely Integrity, Privacy and Confidentiality with key reuse 1.

When the target rate of 10,000 was not sustainable, the distributed queue manager saw a backlog of messages building up on the queue. In addition, the time spent on the transmission queue (XQTIME) is significantly higher when sending messages to the AMS message channel intercept-enabled queue manager.

To prevent the putting application failing, 2 changes were made:

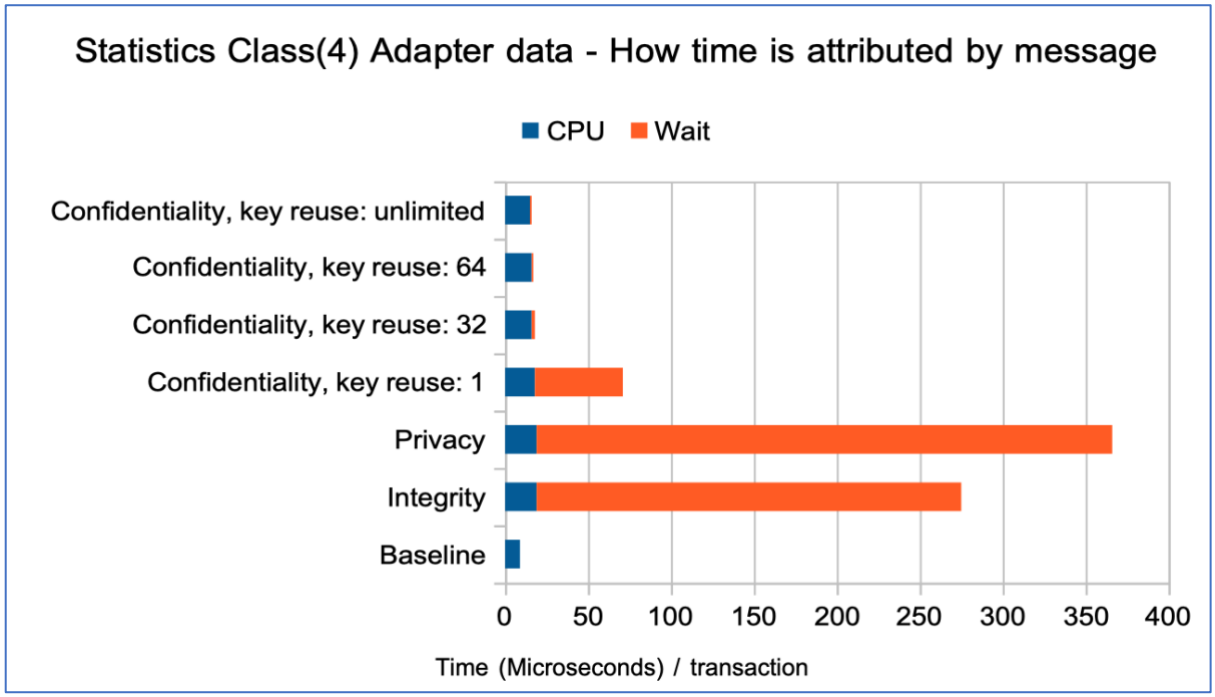
1. The MAXDEPTH attribute on the queue on the distributed queue manager was altered to have a higher value than the default of 5000.
2. The application was altered to wait-then-retry when the MQRC_Q_FULL was returned from the MQPUT call.

The backlog of messages on the distributed queue manager was due to the increased time in the adapter task on the AMS-enabled queue manager, whilst protecting the inbound message.

A secondary affect occurred on the z/OS queue manager for the Confidentiality configuration where key reuse of 1 was specified. In this instance, the rate at which the messages were encrypted and put to the queue out-paced the rate at which the getting task was able to remove the messages from the queue. This resulted in page set I/O, and

eventually the arriving messages' put rate was slowed to allow for immediate writes to page set. This could have been alleviated with larger buffer pools on the z/OS queue manager.

The second chart represents the class(4) statistics data for the adapter task for each configuration:



In the above chart, the integrity, privacy and confidentiality with key-reuse 1 configurations show significant time in wait-state. This time spent waiting means the adapter task is blocked from processing other work.

These are the same 3 configurations that were unable to sustain 10,000 messages per second.

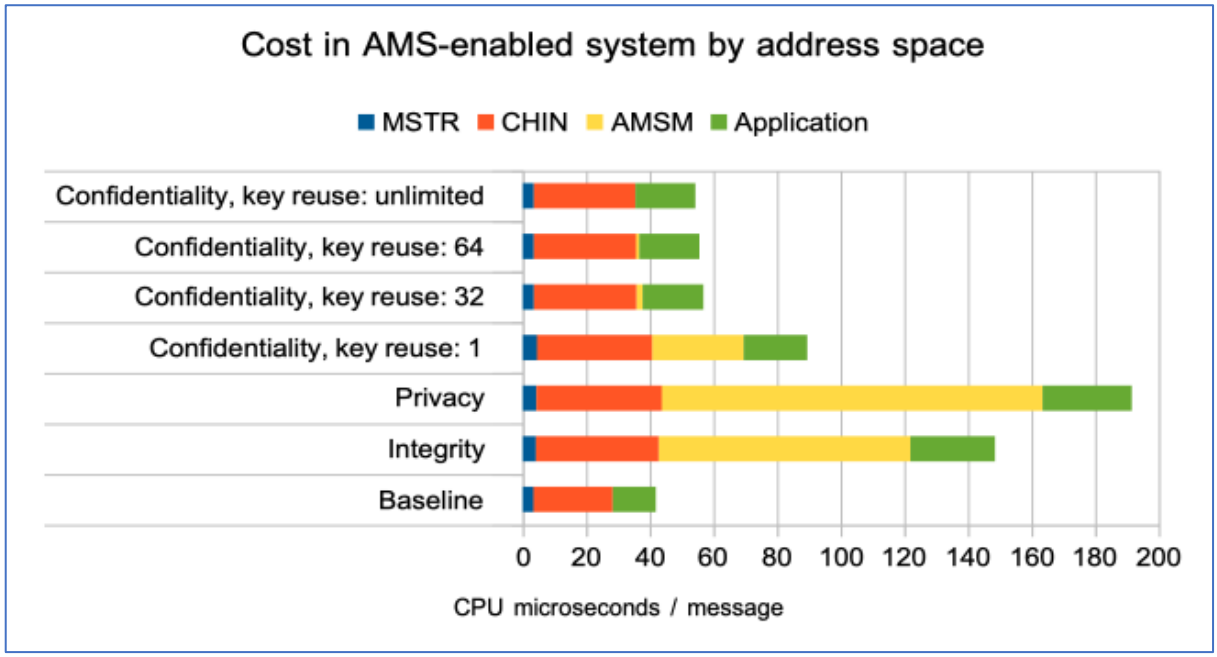
The wait time is calculated from the average elapsed time minus the average CPU time per adapter request.

In this scenario, for the *privacy* and *integrity* configurations, this wait time is largely spent in the cryptographic certificate processing, some of which is charged to the AMSM address space. The *confidentiality key reuse 1* configuration sees wait time, partly due to the immediate writes to page set and partly due to waiting for cryptographic certificate processing.

Note that encryption is performed on CPACF (CP Assist for Cryptographic Functions) and is recorded as CPU time, which is why the wait time on the confidentiality configurations reduces as the key reuse increases.

The final chart in this section shows the cost per message in the MQ address spaces – MSTR, CHIN and AMSM, as well as the cost in the getting application.

Note that the getting application contains minimal processing apart from the MQGET, so the impact of AMS protection appears more significant than in an application where MQ forms 10% of the processing.



For the 3 configurations where the target rate of 10,000 messages per second was not achieved, the queue manager and channel initiator cost per message are higher than when the target rate was achieved. This is in part due to the address spaces not being driven as efficiently at the lower messaging rate and in part due to the additional cost of AMS protection, amounting to 10-14 CPU microseconds per message.

When the channel initiator is processing AMS Confidentiality messages at the same rate as unprotected messages, there is an increase in cost per message of 7 CPU microseconds.

The application address space costs were impacted by the type of protection:

- Integrity added 13 CPU microseconds per MQGET
- Privacy added 14 CPU microseconds per MQGET
- Confidentiality added 5 CPU microseconds per MQGET.

Summary

AMS Interception on server to server message channels provides a solution to a specific problem:

- Ensuring an Enterprise or line of business can protect their MQ data at rest without mandating their partner(s), whether internal or external, implement the same quality of protection, or at the same point in time.

The impact of AMS Interception on z/OS will vary depending on whether you are currently AMS-enabled or not.

It is worth considering whether the increased flexibility from AMS Interception on z/OS outweighs the lower AMS end-to-end protection costs.

Implementing AMS Interception on z/OS may require reviewing your CHIADAPS and CHIDISPS settings on the proposed queue managers in order to minimize the impact on any non-AMS Interception on z/OS workloads.

If the channel initiator is **only** processing AMS Interception on z/OS-type work, increasing the number of adapter tasks may result in increased local lock time rather than improving throughput. However if the channel initiator is processing a mixture of AMS Interception on z/OS-type work, AMS end-to-end work or non-AMS work, increasing the number of adapter tasks may benefit the overall throughput.

Using AMS Interception on z/OS may increase the load on your cryptographic hardware – review the RMF Cryptographic report to determine there is sufficient capacity available for the expected increase in cryptographic work.

Applying AMS protection over server-to-server channels, whether in an end-to-end or a message channel interception configuration, will add both cost and latency to the end-to-end transaction. In a system that is constrained for CPU or cryptographic resource, the use of AMS Interception on z/OS could be a significant factor in a change in the behavior of the workload.

Given the impact that AMS protection over server-to-server message channels can have on throughput and latency, it is always worth reviewing your settings for:

- Maximum queue depth
- Maximum message length – AMS protected messages will be larger than their unprotected counterparts.
- Expiry – with the increased latency from AMS Interception on z/OS, are messages going to be expired before they can reach their target destination?

Finally, ensure the user ID for the channel initiator applying or removing AMS protection is authorized to perform this processing.

Appendix A – Environment under test

System configuration

IBM MQ Performance sysplex running on z14 (3906-7E1) configured thus:
LPAR A: 1-32 dedicated CP processors, 128GB of real storage
LPAR B: 1-10 dedicated CP processors, plus 1 zIIP, 32GB of real storage.

Default Configuration:
3 dedicated processors on each LPAR, where each LPAR running z/OS v2r3 FMID HBB77B0.

Coupling Facility:

- Internal coupling facility with 4 dedicated processors.
- Coupling Facility running CFCC level 23 service level 00.13.
- Dynamic CF Dispatching off.
- 3 x ICP links between each z/OS LPAR and CF.

DASD:

- FICON Express 16S connected DS8870.
- 4 dedicated channel paths (shared across sysplex).
- HYPERPAV enabled.

System settings:

- zHPF disabled by default.
- HIPERDISPATCH enabled by default.
- LPARs A and B configured with different subnets such that tests moving messages over channels send data over 10GbE performance network.
- CryptoExpress6 features configured thus:
    - 1 x Accelerator, shared between LPARs A and B.
    - 2 x Coprocessor on LPAR A.
    - 2 x Coprocessor on LPAR B.

IBM MQ trace status:

- TRACE(GLOBAL) disabled.
- TRACE(CHINIT) disabled.
- TRACE(S) CLASS(1,3,4) enabled.
- TRACE(A) CLASS(1,3,4) enabled.

General information

- Client machines:
    - 2 x IBM SYSTEM X5660 each with 12 x 2.6Ghz processor, 32GB memory.
- BP to AMS-protected Enterprise inbound streaming tests used 10GbE performance network.