UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK

INVINCIBLE IP LLC, Plaintiff v. **SAS INSTITUTE INC.** Defendant	**Case No. 22-cv-04490**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Invincible IP, LLC ("Invincible" or "Plaintiff") files this Complaint for patent infringement against SAS Institute Inc. ("Defendant"), and alleges as follows:

NATURE OF THE ACTION

1. This is an action for patent infringement arising under 35 U.S.C. § 1 *et seq*.

PARTIES

2. Invincible is a limited liability company organized and existing under the laws of the State of Texas with its principal place of business in Plano, Texas.

3. Upon information and belief, Defendant is a corporation organized and existing under the laws of State of North Carolina with a principal place of business at 100 SAS Campus Drive, Cary, North Carolina 27513. Upon information and belief, Defendant may be served through its registered agent, Corporation Service Company, at 80 State Street, Albany, New York 12207.

JURISDICTION AND VENUE

4. This Court has original jurisdiction over the subject matter of this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

5. On information and belief, Defendant is subject to personal jurisdiction of this Court based upon it having regularly conducted business, including the acts complained of herein, within the State of New York and/or deriving substantial revenue from goods and services provided to individuals in the State of New York and in this District.

6. Venue is proper in this District under 28 U.S.C. § 1400(b) because Defendant has committed acts of infringement and has a regular and established place of business in this judicial district.

IDENTIFICATION OF THE ACCUSED INSTRUMENTALITY

7. Defendant provides for its customers use SAS Institute ("the Accused Instrumentality").

COUNT I (Infringement of U.S. Patent No. 8,954,993)

8. Invincible incorporates the above paragraphs as though fully set forth herein.

9. Plaintiff is the owner, by assignment, of U.S. Patent No. 8,954,993 ("the '993 Patent"), entitled LOCAL MESSAGE QUEUE PROCESSING FOR CO-LOCATED WORKERS, which issued on February 10, 2015. A copy of the '993 Patent is attached as Exhibit 1.

10. The '993 Patent is valid, enforceable, and was duly issued in full compliance with Title 35 of the United States Code.

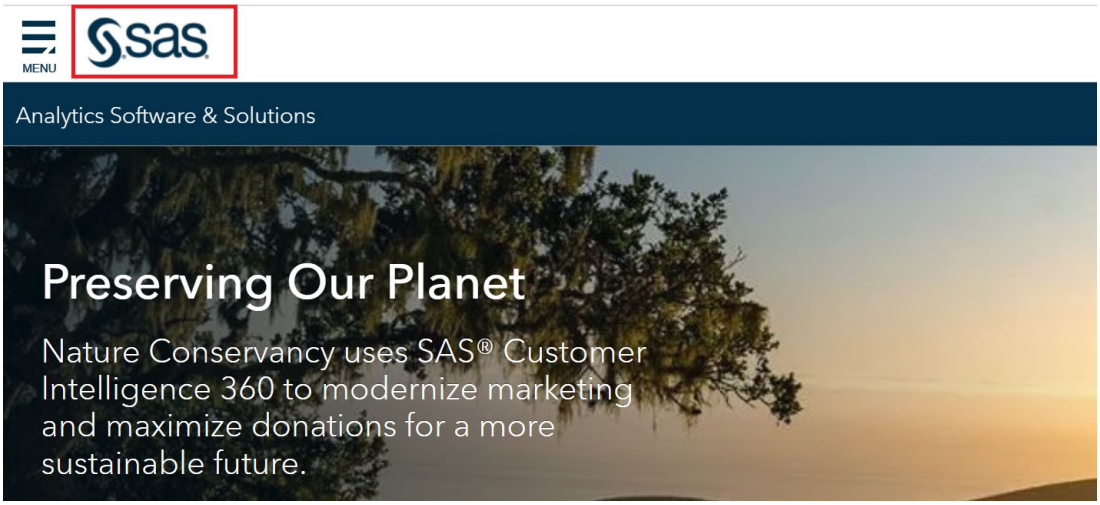
11. Defendant has been and is now infringing one or more claims of the '993 Patent under 35 U.S.C. § 271 by making, using, selling, and offering to sell the Accused Instrumentality in the United States without authority.

12. Claim 1 of the '993 Patent recites:

> 1. A method to locally process queue requests from co-located workers in a datacenter, the method comprising:
>
> detecting a producer worker at a first server sending a first message to a datacenter queue at least partially stored at a second server;
>
> storing the first message in a queue cache at the first server, wherein the queue cache includes one of a copy and a partial copy of the datacenter queue;
>
> detecting a consumer worker at the first server sending a message request to the datacenter queue;
>
> providing the stored first message to the consumer worker in response to the message request;
>
> receiving a signal from a command channel associated with the datacenter queue; and
>
> modifying the stored first message in response to receiving the signal.

13. More particularly, Defendant infringes at least claim 1 of the '993 Patent.

14. On information and belief, Defendant makes, uses, sells, and offers to sell the Accused Instrumentality, which practices a method to locally process queue requests from co-located workers in a datacenter.



Application Messaging Prev | Next | Contents

Application Messaging Overview

Application messaging architectures provide a platform that supports interoperability among loosely coupled applications over a message passing bus. When the targeted scope of interoperability is broad (for example, spanning multiple application systems and organizational boundaries), application messaging architectures might be required. This is because the likelihood of conformance in the software implementation base (for example, the selected distributed object standard) across the set of participating applications is diminished. Additionally, the set of participating applications can exhibit asynchronous, disconnected operation, executing with no direct point-to-point communication session, yet requiring guaranteed fulfillment of requests for service or event delivery.

This degree of operational heterogeneity introduces several requirements that are reflected in the application messaging infrastructure. Heterogeneity in the implementation base of the various applications (including perhaps, retrofitted legacy applications) suggests a need for a reasonably non-intrusive integration mechanism. The semantics of application messaging satisfy this need, generally expressing open, close, send, and receive functionality with flexible application-defined message structures. Heterogeneity with respect to the asynchronous, disconnected execution and notification modes of end-point participants introduces requirements for service qualities that include routing, assured just-once delivery, and retained sequencing. The architecture that has emerged within commercial application messaging products to express these quality-of-service properties is *store-and-forward queuing*.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.

What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

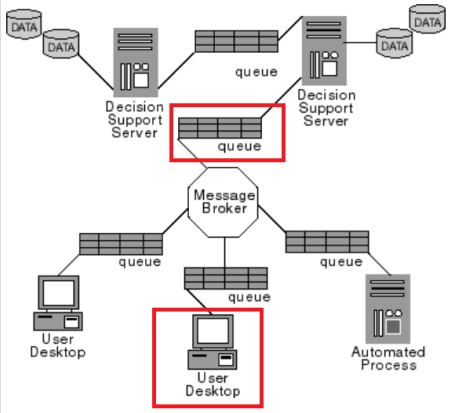
SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

Create a Message Queue Server and Register Queues

To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.



15. On information and belief, the Accused Instrumentality practices detecting a producer worker (e.g. producer application) at a first server (e.g. SAS server), sending a first message to a datacenter queue (e.g. SAS message queue) at least partially stored at a second server (e.g. Message Queue server).

What Are Message Queues?

Producer Worker

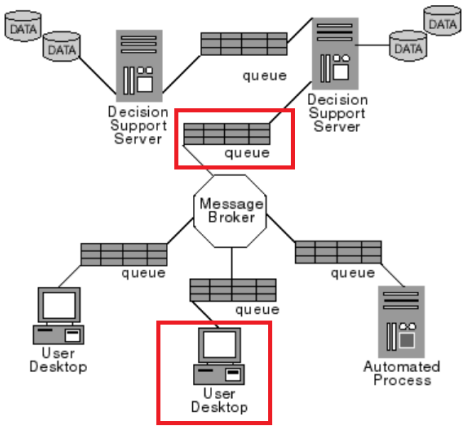
Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.



Create a Message Queue Server and Register Queues

To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

16. On information and belief, the Accused Instrumentality practices storing the first message in a queue cache at the first server (e.g., SAS server) wherein the queue cache includes one of a copy and a partial copy of the datacenter queue (e.g., SAS message queue). Further, on information and belief, the first server with the producer stores messages in a queue cache at the first server.

What Are Message Queues?

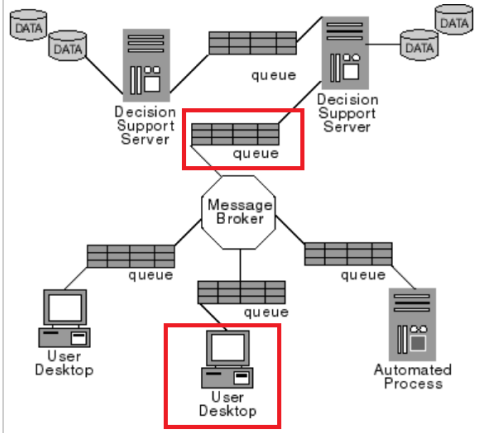
Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.



Create a Message Queue Server and Register Queues

To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

17. On information and belief, the Accused Instrumentality practices detecting a consumer worker (e.g., a consumer application) at the first server (e.g., SAS server), sending a message request (e.g. request messages from the queue) to the datacenter queue (e.g. SAS message queue).

What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Consumer Worker

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

18. On information and belief, the Accused Instrumentality practices providing the stored first message to the consumer worker (e.g., Consumer application) in response to the message request (e.g. request messages from the queue).

What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

19. On information and belief, the Accused Instrumentality practices receiving a signal (e.g. Delete request) from a command channel associated with the datacenter queue (SAS message queue).

To modify the metadata definitions of queues and queue managers, use the New Server Wizard in SAS Management Console to delete the existing definitions and create new definitions.

Before you delete metadata, you might want to run impact analysis and reverse impact analysis to see the jobs that include the queues. To run impact analysis or reverse impact analysis, follow these steps:

1. In SAS Data Integration Technologies, click the **Inventory** tab and expand **Message Queues**.
2. Right-click a message queue and select **Analyze**.
3. In the Analysis window, examine the graphical displays of impact analysis and reverse impact analysis to see how the message queue is used in existing jobs.

You can delete the message queue if the analysis diagrams show no connections to existing jobs or other metadata objects. If the diagrams show connections, edit or remove the metadata objects before you delete the message queue.

To delete metadata for queues and queue managers, right-click the object in the Server Manager of SAS Management Console and select **Delete**. Use the third-party software to delete the physical queue objects.

attr

Specifies a delete attribute. The following attributes are valid:

DELETE

Specifies that the queue is to be deleted after it successfully closes, but only if there are no messages on the queue. This attribute is supported with MQSeries only. It is not supported with MSMQ because there is no way to programmatically determine the depth of the queue. It is not supported with Rendezvous because Rendezvous handles this function internally.

DELETE_PURGE

Causes the queue to be deleted, even if the queue depth is greater than zero. This attribute is supported with MQSeries, MQSeries C, MSMQ, and Rendezvous-CM. It is not supported with Rendezvous because Rendezvous handles this function internally. If you are using Rendezvous Certified Message Delivery, when you close a listener queue the default setting is for the sender to save messages for persistent messaging. If you do not want messages to be saved by the sender or do not want persistent messaging, specify the DELETE_PURGE attribute when you close the queue. Setting the DELETE_PURGE attribute is the same as setting the cancelAgreements argument on TIBRVCM_CANCEL(TRUE).

20. On information and belief, the Accused Instrumentality practices modifying the stored first message (e.g., deleting the queue containing the first message) in response to receiving the signal (e.g. Delete request).

To modify the metadata definitions of queues and queue managers, use the New Server Wizard in SAS Management Console to delete the existing definitions and create new definitions.

Before you delete metadata, you might want to run impact analysis and reverse impact analysis to see the jobs that include the queues. To run impact analysis or reverse impact analysis, follow these steps:

1. In SAS Data Integration Technologies, click the **Inventory** tab and expand **Message Queues**.
2. Right-click a message queue and select **Analyze**.
3. In the Analysis window, examine the graphical displays of impact analysis and reverse impact analysis to see how the message queue is used in existing jobs.

You can delete the message queue if the analysis diagrams show no connections to existing jobs or other metadata objects. If the diagrams show connections, edit or remove the metadata objects before you delete the message queue.

To delete metadata for queues and queue managers, right-click the object in the Server Manager of SAS Management Console and select **Delete**. Use the third-party software to delete the physical queue objects.

attr

Specifies a delete attribute. The following attributes are valid:

DELETE

Specifies that the queue is to be deleted after it successfully closes, but only if there are no messages on the queue. This attribute is supported with MQSeries only. It is not supported with MSMQ because there is no way to programmatically determine the depth of the queue. It is not supported with Rendezvous because Rendezvous handles this function internally.

DELETE_PURGE

Causes the queue to be deleted, even if the queue depth is greater than zero. This attribute is supported with MQSeries, MQSeries C, MSMQ, and Rendezvous-CM. It is not supported with Rendezvous because Rendezvous handles this function internally. If you are using Rendezvous Certified Message Delivery, when you close a listener queue the default setting is for the sender to save messages for persistent messaging. If you do not want messages to be saved by the sender or do not want persistent messaging, specify the DELETE_PURGE attribute when you close the queue. Setting the DELETE_PURGE attribute is the same as setting the cancelAgreements argument on TIBRVCM_CANCEL(TRUE).

21. Plaintiff has been damaged by Defendant's infringing activities.

COUNT II (Infringement of U.S. Patent No. 9,479,472)

22. Invincible incorporates the above paragraphs as though fully set forth herein.

23. Plaintiff is the owner, by assignment, of U.S. Patent No. 9,479,472 ("the '472 Patent"), entitled LOCAL MESSAGE QUEUE PROCESSING FOR CO-LOCATED WORKERS, which issued on October 25, 2016. A copy of the '472 Patent is attached as Exhibit 2.

24. The '472 Patent is valid, enforceable, and was duly issued in full compliance with Title 35 of the United States Code.

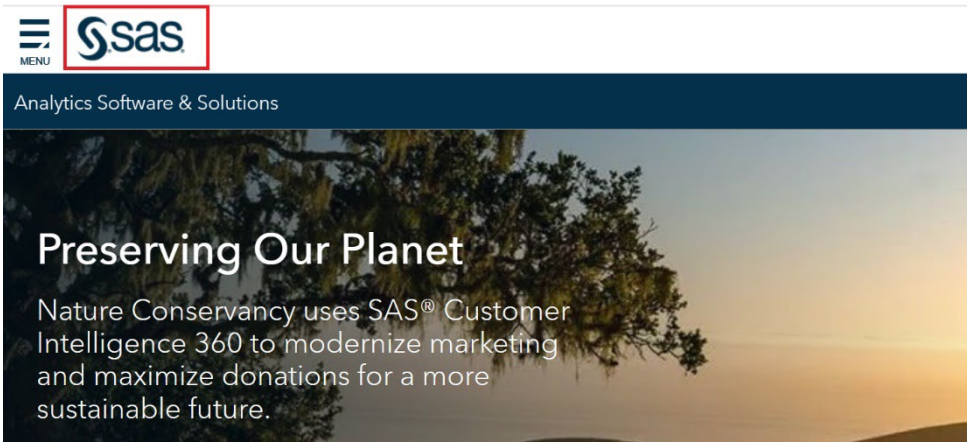
25. Defendant has been and is now infringing one or more claims of the '472 Patent under 35 U.S.C. § 271 by making, using, selling, and offering to sell the Accused Instrumentality in the United States without authority.

26. Claim 1 of the '472 Patent recites:

> 1. A method to locally process queue requests from co-located workers in a datacenter, the method comprising:
>
> detecting a producer worker at a first server, wherein the producer worker sends a message to a datacenter queue at least partially stored at a second server;
>
> storing the message in a queue cache at the first server;
>
> detecting a consumer worker at the first server, wherein the consumer worker sends a message request to the datacenter queue; and
>
> providing the message to the consumer worker in response to the message request.

27. More particularly, Defendant infringes at least claim 1 of the '472 Patent.

28. On information and belief, Defendant makes, uses, sells, and offers to sell the Accused Instrumentality, which practices a method to locally process queue requests from co-located workers (e.g, Producer Applications, consumer applications etc. in SAS server) in a datacenter (e.g. SAS datacenter).



Application Messaging Overview

Application messaging architectures provide a platform that supports interoperability among loosely coupled applications over a message passing bus. When the targeted scope of interoperability is broad (for example, spanning multiple application systems and organizational boundaries), application messaging architectures might be required. This is because the likelihood of conformance in the software implementation base (for example, the selected distributed object standard) across the set of participating applications is diminished. Additionally, the set of participating applications can exhibit asynchronous, disconnected operation, executing with no direct point-to-point communication session, yet requiring guaranteed fulfillment of requests for service or event delivery.

This degree of operational heterogeneity introduces several requirements that are reflected in the application messaging infrastructure. Heterogeneity in the implementation base of the various applications (including perhaps, retrofitted legacy applications) suggests a need for a reasonably non-intrusive integration mechanism. The semantics of application messaging satisfy this need, generally expressing open, close, send, and receive functionality with flexible application-defined message structures. Heterogeneity with respect to the asynchronous, disconnected execution and notification modes of end-point participants introduces requirements for service qualities that include routing, assured just-once delivery, and retained sequencing. The architecture that has emerged within commercial application messaging products to express these quality-of-service properties is *store-and-forward queuing*.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.

What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

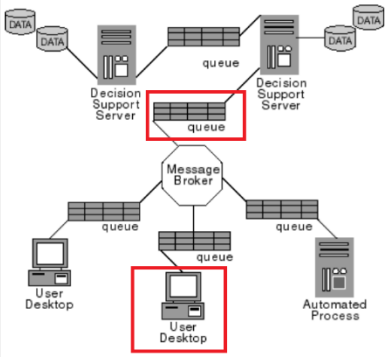
SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

Create a Message Queue Server and Register Queues

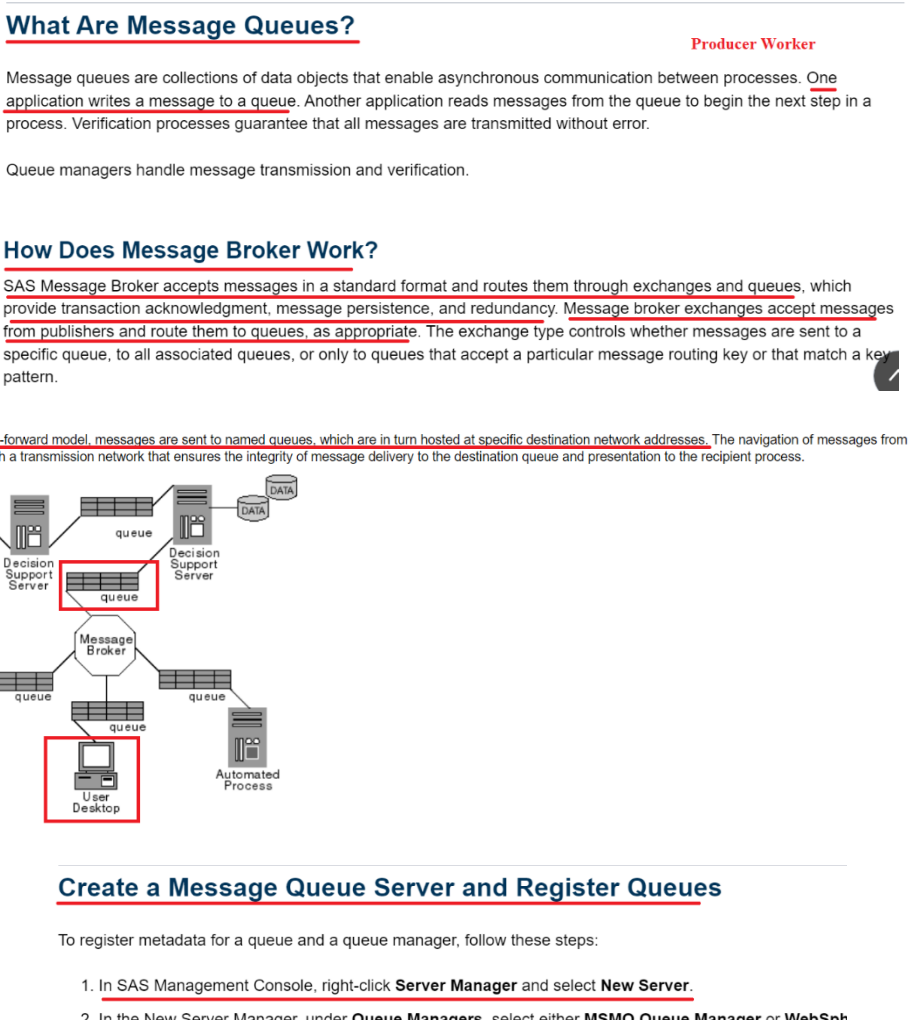
To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.



29. On information and belief, the Accused Instrumentality practices detecting a producer worker (e.g., producer application) at a first server (e.g. SAS server), wherein the producer worker (e.g., producer application) sends a first message to a datacenter queue (e.g. SAS message queue) at least partially stored at a second server (e.g. Message Queue server).



What Are Message Queues?

Producer Worker

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.

Create a Message Queue Server and Register Queues

To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

30. On information and belief, the Accused Instrumentality practices storing the first message in a queue cache at the first server (e.g., SAS server).

What Are Message Queues?

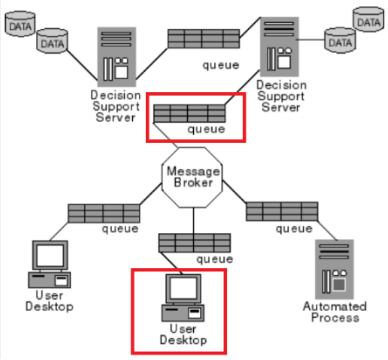
Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.

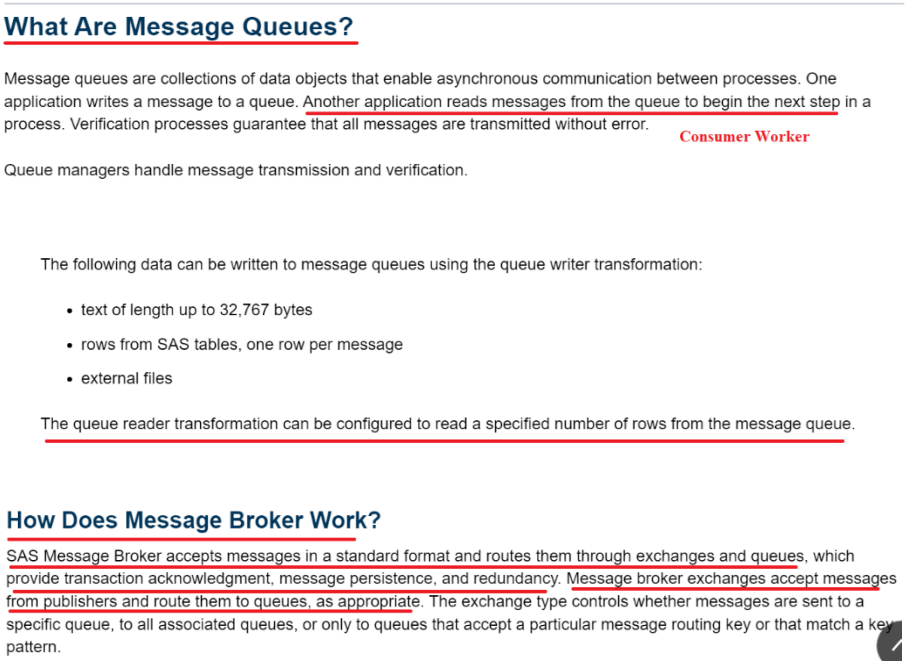


Create a Message Queue Server and Register Queues

To register metadata for a queue and a queue manager, follow these steps:

1. In SAS Management Console, right-click **Server Manager** and select **New Server**.
2. In the New Server Manager, under **Queue Managers**, select either **MSMQ Queue Manager** or **WebSph**

31. On information and belief, the Accused Instrumentality practices detecting a consumer worker (e.g., a consumer application) at the first server (e.g., SAS server), wherein the consumer worker (e.g., a consumer application) sends a message request (e.g. request messages from the queue) to the datacenter queue (e.g. SAS message queue).



What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Consumer Worker

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

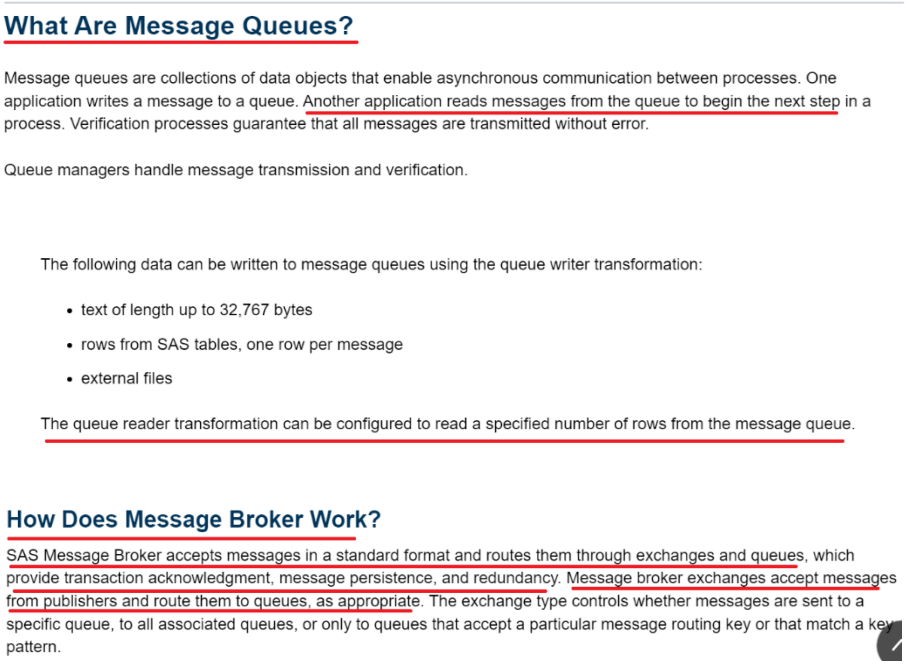
- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

32. On information and belief, the Accused Instrumentality practices providing the stored first message to the consumer worker(e.g., Consumer application) in response to the message request (e.g. request messages from the queue).



What Are Message Queues?

Message queues are collections of data objects that enable asynchronous communication between processes. One application writes a message to a queue. Another application reads messages from the queue to begin the next step in a process. Verification processes guarantee that all messages are transmitted without error.

Queue managers handle message transmission and verification.

The following data can be written to message queues using the queue writer transformation:

- text of length up to 32,767 bytes
- rows from SAS tables, one row per message
- external files

The queue reader transformation can be configured to read a specified number of rows from the message queue.

How Does Message Broker Work?

SAS Message Broker accepts messages in a standard format and routes them through exchanges and queues, which provide transaction acknowledgment, message persistence, and redundancy. Message broker exchanges accept messages from publishers and route them to queues, as appropriate. The exchange type controls whether messages are sent to a specific queue, to all associated queues, or only to queues that accept a particular message routing key or that match a key pattern.

33. Plaintiff has been damaged by Defendant's infringing activities.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff respectfully requests the Court enter judgment against Defendant:

1. declaring that Defendant has infringed the '993 Patent;
2. awarding Plaintiff its damages suffered as a result of Defendant's infringement of the '993 Patent;
3. declaring that Defendant has infringed the '472 Patent;
4. awarding Plaintiff its damages suffered as a result of Defendant's infringement of the '472 Patent;
5. awarding Plaintiff its costs, attorneys' fees, expenses, and interest; and
6. granting Plaintiff such further relief as the Court finds appropriate.

JURY DEMAND

Plaintiff demands trial by jury, under Fed. R. Civ. P. 38, on all issues so triable.

Dated: May 31, 2022

Respectfully submitted,

/s/ Nicholas Loaknauth

Loaknauth Law, P.C.
Nicholas Loaknauth
SDNY Bar No. NL0880
1460 Broadway
New York, New York 10036
Telephone: (212) 641-0745
Email: nick@loaknauth.com

Together with:

Raymond W. Mort, III (*pro hac* to be filed)
Texas State Bar No. 00791308
raymort@austinlaw.com

THE MORT LAW FIRM, PLLC
100 Congress Ave, Suite 2000
Austin, Texas 78701
Tel/Fax: (512) 865-7950

ATTORNEYS FOR PLAINTIFF