Push and Pull Message Delivery in the PSB Architecture.

Discussion and Recommendations.

Version: 1.7

Date: 10 September 2004.

Sean McGrath.

Reach.

Version History		
Date	**Version**	**Comments**
08/06/04	0.9	First version, incorporating material from other sources.
09/06/04	1	Incorporated initial obs from PatC. Some new industry examples. Executive summary.
09/06/04	1.1	Fixed typos.
10/06/04	1.2	Added section on shared infrastructure in a push model. Added Adapter Equality principle to Appendix 1.
19/07/04	1.3	Added section on polled pull versus blocked pull. Incorporated obs from PatC.
08/08/04	1.4	Modified a lot of the wording for clarity. Cleaned up the push/pull terminology using "transmission" and "delivery" qualifiers. Added more material to hop 2. Additional material in hybrid push/pull delivery section (now referred to as local push transport adapter). Re-arranged some sections. Added section on ACK/NACK messages. Fixed some typos.
23/08/04	1.5	Incorporated feedback from PatM.
24/08/04	1.6	Incorporated feedback from PatM
10/09/04	1.7	Minor typographical changes.

Changes since last version
See above.

Known Omissions
More examples of industry practice under preparation.

Table of Contents

1 Introduction and Executive Summary

This document outlines at business and technological levels, the rationale behind the favouring of so-called "pull delivery" over "push delivery" for the delivery of messages from the PSB to *agency-hosted* PSB services. The pros and cons of each model are presented and a list of recommendations is provided.

The agency-hosted versus PSB-hosted distinction is a vital one in this document. The majority of the reasons for favouring pull delivery over push-delivery are related to organisational boundaries and the complexities added by wide area networks such as the Government VPN (GVPN). These are greatly diminished when services are co-located with the PSB – both organisationally and physically.

1.1 A simple message exchange

There are two sides to every simple message exchange in the PSB architecture. An *originating service* transmits a message and a *destination service* receives the message.

The PSB favoured model for this message exchange is that:

Originating services *push* messages to the PSB (push transmission)

Receiving services *pull* messages from the PSB (pull delivery)

To date, the favouring of push-based transmission from originating services has not been an issue. Consequently, this document is primarily focused on the *delivery* side of the message exchange.

1.2 Pros of the push delivery model

- It appears to add to the PSB value proposition from an agency perspective e.g. "the PSB brings messages right into my systems with the push delivery model".
- The push delivery model is the default model in some popular commercial messaging products.

1.3 Cons of push delivery model

- The push delivery model creates significant organisational responsibility issues for Reach and agencies. In order to be robust, part of the agency infrastructure will need to be under Reach's control. This is because it needs to be tightly integrated with the infrastructure on Reach's side that is pushing the messages.

  The question of responsibility – especially in the event of operational issues

arising – becomes complex. With a pull delivery model on the other hand, organisational boundaries are crisp and clear. There is no need for any agency infrastructure to be under Reach's control. Responsibility for getting messages from the PSB lies squarely with the agencies.

- The push delivery model requires a large agency footprint. This is because equipment to receive pushed messages needs to reside within the agency domain and yet be visible to the PSB.

  Typically, this involves adding machines to network operations rooms, modifying firewalls etc. The pull delivery model by contrast, has a small agency footprint. It does not require any visibility of the agency domain by the PSB.
- The push delivery model increases detrimental technology coupling, particularly in the critical area of temporal coupling. Machines need to be in synchronous lock-step in order to function correctly. i.e. the machine receiving messages has to be available when the PSB attempts to push messages, otherwise error conditions arise.
- In a push delivery model, the PSB becomes a proactive rather than a reactive actor in the business processes of agencies. Again, this creates organisational boundary issues. For example, if the PSB fails to bring new messages into an agency's infrastructure when it expect them, whose fault is it? With a push delivery model, the PSB is central to the business processes of agencies in a way that would make the PSB a focal point for all operational issues in inter-agency business processes.

  Examples:

  - "There is a problem with the PSB because it did not push my message through to me on time."
  - "There is a problem with the PSB because I have no way of knowing if/when it will deliver me messages."
  - "There is a problem with the PSB because Reach has not sent an engineer out to fix the problem with Reach's box in my infrastructure."
  - and so on.
- The large agency footprint of the push-based delivery model creates a barrier to entry for agencies into PSB messaging. The need to configure shared infrastructure is time consuming and costly. Ongoing maintenance of shared infrastructure is an issue. Who pays for it? Who fixes it when it goes wrong? Where does the fault lie? Reach or agency? and so on.

1.4 Summary of Recommendations

The main recommendations in this document are:

1. That "pull delivery" be the default message delivery pattern of the PSB for agency hosted services.
2. That "pull delivery" be implemented by all transport adapters in line with the "all adapters are equal" principle of the TDS [TDS].
3. That agencies be fully appraised of the business and technical disadvantages – both from their perspective and from Reach's perspective - of the alternative "push delivery" model.
4. That vendors of technologies likely to feature in PSB Transport Adapters be appraised of the architecture and invited to provide configurations of their systems that work "out of the box" with the "pull delivery" model.
5. That the sharing of any operational issues surrounding the GVPN be clearly set out in the T&C documents between Reach and agencies.

2 A high level explanation of the PSB messaging architecture.

2.1 Overview of the PSB architecture.

A high-level view of the PSB architecture is shown in Figure 1.

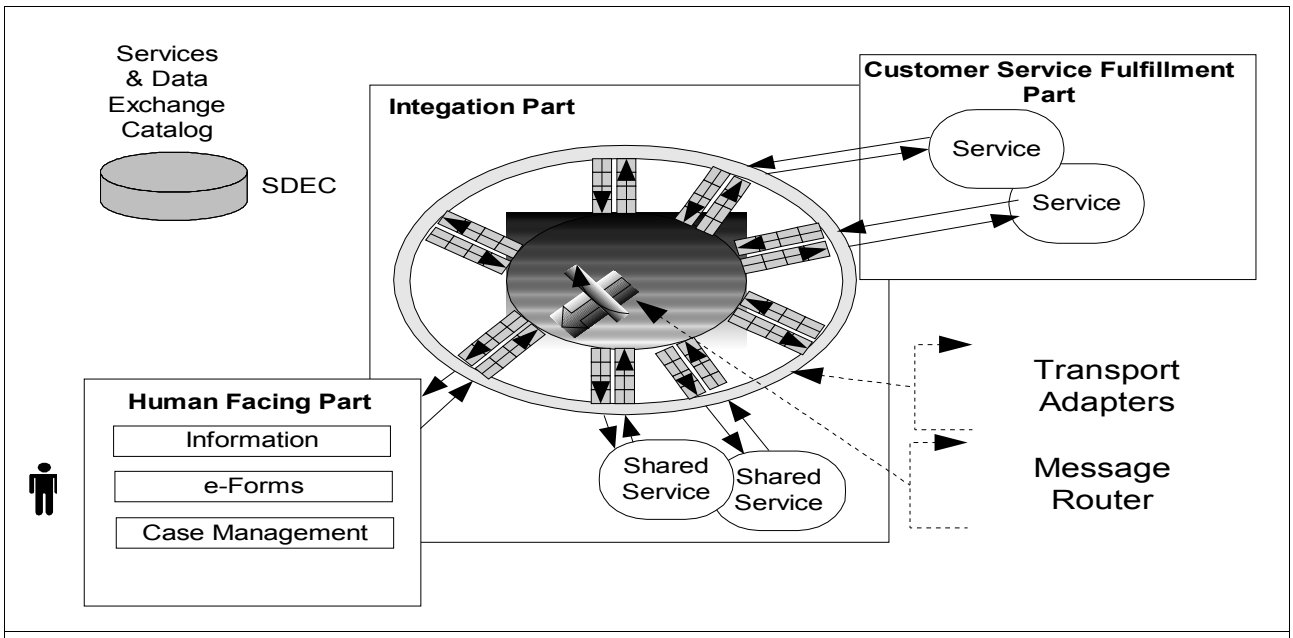

Figure 1: High-level view of the PSB architecture.

Agencies (or more accurately, the services hosted by agencies) send messages (typically XML) to each other using the PSB integration framework.

The mechanics of this for two services, A and B, sending messages to each other is shown in Figure 2.

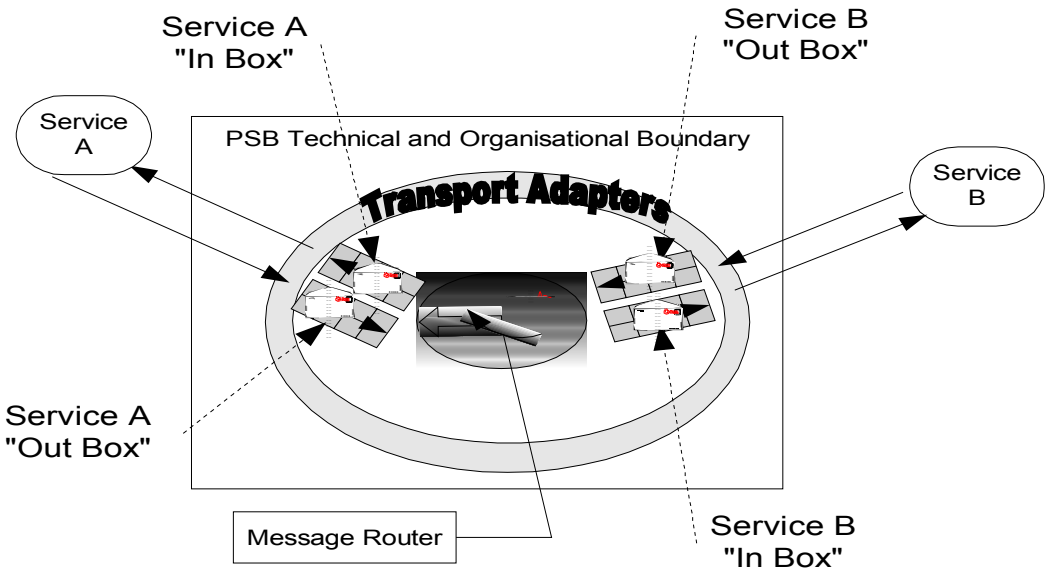

Figure 2: Two services A and B exchanging messages via the PSB.

A useful way to conceptualise the message transfer is to think of each service as having its own "in-box" and "out-box" as shown in Figure 2.

A service that wishes to send a message to another service does so by dropping a message in the out-box maintained for it by the PSB.

The core of the PSB essentially acts as a courier – moving messages from the out-boxes of message senders to the in-boxes of message receivers.

N.B.: the in-boxes and out-boxes for services are housed, managed and maintained by the PSB.

This model completely decouples service A from service B. These services are completely autonomous, yet they can communicate in the heterogeneous, multi-agency world of government.

This loose coupling is a keystone of the PSB and it is a key enabler of robust, scalable web service-based systems [Helland].

2.2 Request messages and response messages

In the PSB architecture it is common to set up a message choreography that involves positive/negative acknowledgement of business level messages. These are referred to as ACKs and NACKs respectively.

An important aspect of the asynchronous messaging substrate underlying the

PSB is that ACKs and NACKs are completely separate messages from the messages that trigger them. An agency sending a message to the PSB will receive any associated ACK/NACK messages in exactly the same way it would receive any other message, i.e., by communicating with a transport adapter. Consequently, the same push/pull issues arise for ACK/NACK messages as apply for ordinary messages.

Note that the PSB will provide a "synch-over-asynch" transport adapter that will allow agencies to treat "Send a message, get an ACK/NACK back." as a single atomic action. Behind the scenes, the PSB achieves this with two back-to-back messages. This form of synchronous exchange is only suitable for use with idempotent services.

3 An analysis of the three hops involved in each message exchange

Figure 3 illustrates a specific message exchange scenario in which Service A sends a message to Service B.

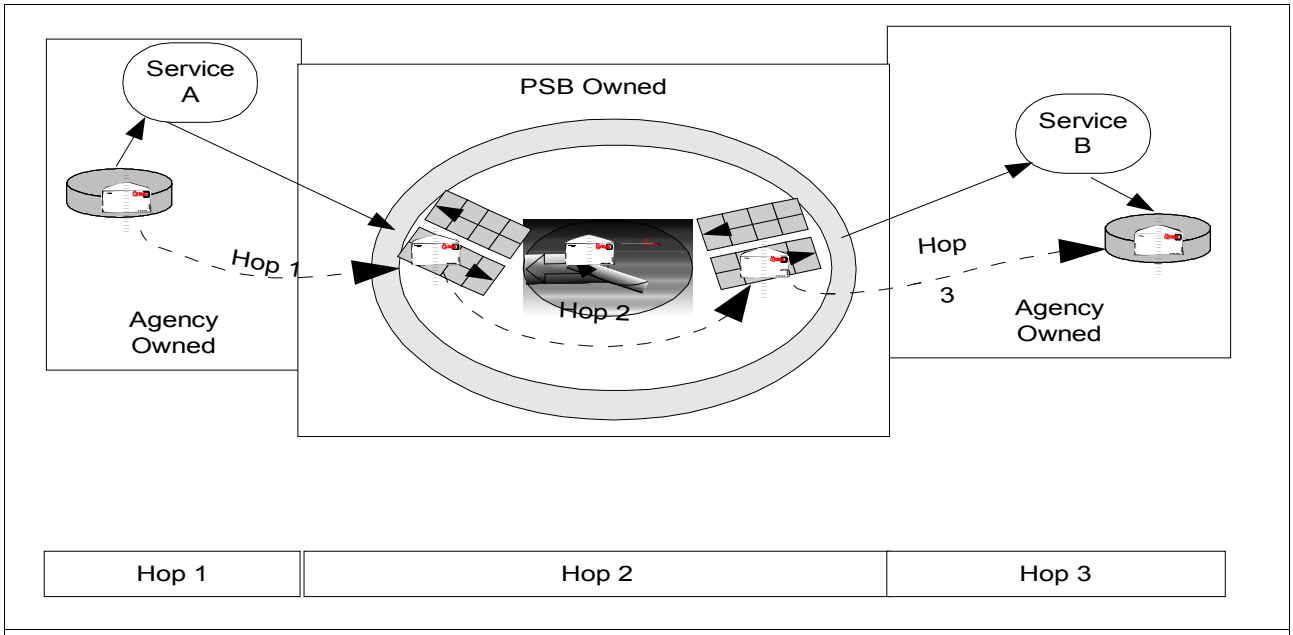

Figure 3: Service A sends a message to service B.

The message transfer takes place in three "hops":

Hop 1 – The message moves from service A (agency owned) into the PSB where it is deposited in the "out-box" for service A.

Hop 2 – The message moves from the "out box" of service A to the "in-box" of service B.

Hop 3 – The message moves from the "in-box" for service B out to Service B (agency owned).

3.1 The PSB messaging value proposition for inter-agency message exchange.

The value proposition of the PSB for this message exchange is as follows:

1. The PSB *guarantees* that the message from A, once successfully handed over to the PSB in Hop 1, will be routed to an "out-box" that the PSB maintains on behalf of service B.
2. The PSB *guarantees* that any messages sent to B, from any other service, will be safely stored in B's outbox until such time as service B collects the messages.
3. The PSB *guarantees* that messages will not be lost within the PSB infrastructure – they are guaranteed to be moved from a producer "in-box" to a consumer "out-box".
4. The PSB *guarantees* that messages will not be unknowingly modified within the PSB infrastruture (tamper evidence).
5. The PSB *guarantees* that messages will not be duplicated. ie. The message from A will be stored in the "out-box" for B *exactly once*.

4 Implementation options for the three messaging hops

This document details each of the three hops and discuss the pros and cons of "push/pull delivery" and "push/pull transmission" implementations from a PSB architectural principle perspective. These principles are outlined in the TDS [TDS].

Specifically this document details push/pull implications for:

- Organisational responsibility
- Agency footprint. See TDS section 3.2.6 (Reproduced in Appendix 1)
- Loose coupling. See TDS section 3.2.1 (Reproduced in Appendix 1)

4.1 Hop 1 – From Agency Service to the PSB

There are two options for this hop. Either:

- The message transfer is initiated by the service (push transmission), or
- The message transfer is initiated by the PSB (pull transmission)

4.1.1 Option 1 – the message transfer is initiated by the service (push transmission)

In this model, the service contacts the PSB and says, in effect, "I have a message to send. Please take it from me and deliver it safely to its destination."

4.1.1.1 Pros of push transmission

The **organisational responsibility** boundaries are clear. It is the responsibility of the agency that owns the service, to get the message into the PSB in the first instance. It is not the PSB's responsibility to come looking for messages. (Analogy: the postal company will route what it finds in the postbox. It is not the responsibility of the postal company to go around asking everyone if they have any messages to send.).

There is a small **agency footprint**. The agency does not need any special technology setup to host the service. This is because all traffic is initiated from the agency side and can, for example, be routed out over already open ports such as HTTP port 80. There is no need for supporting infrastructure in the agency's network infrastructure (e.g. no need for a presence in DMZ. No need for changes to firewall configurations etc.)

The small **agency footprint** makes the barrier to entry for agencies to participate in PSB messaging as low as possible.

There is **low coupling**, especially in the critical area of **temporal coupling**. There is no need for the agency to have technology working in lock-step, synchronous fashion with the PSB.

Critically, the PSB is a *reactive* rather than *proactive* actor in the business processes of the agencies. Simply put, agencies do not have to think: "I create a message. Now it is the PSB's responsibility to come and get it.". Instead they think: "I create a message. Now it is *my* responsibility to get it to the PSB.". The former leads to crisper **organisational responsibility** boundaries.

4.1.1.2 Cons of push transmission

Based on a surface analysis, it may appear to some that the push transmission model detracts from the PSB value proposition. e.g. "Why doesn't the PSB make my life easier by coming and getting the messages from me?"

4.1.2 Option 2 – message transfer initiated by the PSB (pull transmission)

In this model, the PSB contacts the service and says, in effect, "Have you any messages you would like me to send?".

4.1.2.1 Pros of pull transmission

Based on a surface analysis, it appears to add to the PSB value proposition. i.e. An agency can simply create messages and leave it to the PSB to come and get them.

4.1.2.2 Cons of pull transmission

Organisational boundaries are complex and cut across existing trust boundaries. In order to collect messages from agencies, the PSB would need open access to part of the agency-controlled infrastructure. The question of ownership and responsibility for that infrastructure becomes complex. If there is a problem getting a message from the agency's side to the PSB's side, whose fault is it? Whose responsibility is it to fix it? Etc.

Agency footprint is large as there needs to be a route whereby the PSB can reach inside the agency's infrastructure and pull messages out. Standard infrastructure practice - in terms of DMZ's, firewalls and exposing Web Services - makes this a non-trivial proposition for agencies.

Temporal coupling is high. There is a time dependency between the creation of a message on the agency side and the PSB "calling in" to collect it. This makes the PSB a proactive rather than a reactive actor in the business processes of the agencies. This raises organisational boundary questions. If the PSB has not called "on time" to collect a message, whose fault is it? The risk here is that the PSB becomes perceived as a brake rather than an accelerator of inter-agency message exchanges.

4.1.3 Recommendation

The pros of Option 1 – **message transfer initiated by the agency in hop 1 (push transmission)** - significantly outweigh the cons and this is the recommended approach.

When sending messages, it is up to the agency to push them to the PSB.

4.2 Hop 2 – PSB Internal

The same push/pull issues arise in Hop 2 as arise in Hops 1 and 3. However, because this hop is completely within the confines of the PSB core, it has no business issues associated with it from an agency perspective.

Note: The remainder of this section can be skipped without loss of continuity. It is provided here for completeness only.

The intuitively simple model for Hop 2 is one in which the routing component is responsible for sweeping through the message queues looking for messages that require routing.

If you think of the router as a "service" it has the choice of using pull delivery or push delivery to receive the messages it needs to route.

The J2EE architecture on which the PSB core is built, provides two pull delivery mechanisms:

(a) blocked JMS read (pull delivery – thread controlled)

(b) message driven beans (pull delivery – appserver controlled)

In (a), a program opens a connection to a message queue and then "listens" for a message. The listen request causes the program to "block" until such time as a message is available, at which point the program resumes execution.

In (b), a program declares an interest in a message queue and tells the appserver what piece of code to invoke when a message is available. The appserver then takes responsibility for listening to the message queue on behalf of the program. When a message arrives, the appserver invokes the piece of code.

In practice, it is likely that the PSB core will make use of both these approaches.

Other integration architectures such as .NET offer similar functionality.

4.2.1 Recommendation

This hop can be implemented using whatever push/pull transmission/delivery models best suits the implementer and the technology stack used for the PSB core.

4.3 Hop 3 – From PSB to Service

There are two options for this hop. Either:

- The message transfer is initiated by the service (pull delivery), or
- The message transfer is initiated by the PSB (push delivery)

4.3.1 Option 1 – message transfer initiated by the service (pull delivery)

In this model, the service contacts the PSB and says, in effect: "Please give me my messages".

4.3.1.1 Pros of pull delivery model

The **organisational responsibility** boundaries are clear. It is the responsibility of the agency to remove messages from the per-service "in-box" that the PSB maintains on its behalf. It is not the PSB's responsibility to ensure that each message makes it into the agency's own infrastructure.

The PSB is reactive rather than proactive in getting the message into the agencies infrastructure. Agencies do not think "I wonder when the PSB will arrive with my messages?". Instead they think "If I want my messages I can go ask for them." and "if I need to know if there any messages waiting for me, I can go ask the PSB".

There is a small **agency footprint**. Agencies do not have to have infrastructure available 24x7 to handle PSB messages. They can start up and close down their own messaging technologies without regard to Reach.

An agency does not need any special technology setup. This is because all traffic is initiated from the agency side and can, for example, be routed out over already open ports such as HTTP port 80. There is no need for supporting infrastructure in the agency's DMZ, no need to change firewall configurations etc.

The pull delivery model decreases detrimental technology coupling, particularly in the critical area of **temporal coupling**. Machines do not need to be in synchronous lock-step in order to function correctly. i.e. the machine pulling messages does not have to be constantly available or timed to correspond to PSB delivery schedules.

Administrative complexity is low. There is no need to maintain PSB infrastructure within the agency's own domain.

If an agency attempts to pull its messages and fails to do so for some reason, the agency will get complete diagnostics from its own network infrastructure as to the cause of the problem. If there is doubt as to where the problem lies, it is a simple matter to ask the PSB if it ever saw an attempted pull. If the answer is no, then the problem is somewhere within the shared GVPN infrastrcture or on the agency side. In a push-delivery model, such diagnostics are more complex because the PSB has very little visibility of the agency infrastructure.

4.3.1.2 Cons of pull delivery model

Based on a surface analysis, this model appears to detract from the PSB value proposition. e.g. Why cannot the PSB deliver messages right into my infrastructure?

Based on a surface analysis the pull model might appear to make the PSB less real-time in its behaviour. Agencies "call and collect" messages rather than receive deliveries of messages. (The pull model does not make the PSB less real-time in its behaviour. See section 5.1 - blocked pull.)

4.3.2 Option 2 – message transfer initiated by the PSB (push delivery)

4.3.2.1 Pros of push delivery

Appears to add to the PSB value proposition. e.g. "The PSB delivers messages right into my infrastructure."

Appears to make the PSB more real-time in its behavior.

4.3.2.2 Cons of push delivery

When the PSB proactively delivers messages by pushing them to the agency, a doubt is created when no messages arrive at the agency side. From the agency's perspective it cannot tell the difference between "no messages for you" and a messaging fault in the PSB. This creates a complex **organisational responsibility** issue.

The **temporal coupling** in this model becomes a significant issue in the event that agency infrastructure is not in a position to receive incoming messages from the PSB. What does the PSB do if it cannot deliver a message? Keep retrying? Move messages to a suspend area and notify an operator? These issues are purely a consequence of the temporal coupling in this model. They cease to be issues with a pull-based delivery model.

Administrative complexity is high because there is a need to maintain infrastructure within the agency's own domain. The organisational issues this raises are non-trivial. Moreover, the level of **temporal coupling** carries with it significant administrative overhead when things go wrong – most notably, when agency infrastructure cannot receive messages.

If the PSB is pushing a message into an agency and there is an error, it is likely that the error report will be of a very general nature e.g.

> *Could not send message to X*.

(where X is some resource exposed by the agency's network infrastructure)

The non-specific nature of the error message is a consequence of the PSB having no visibility beyond the resource exposed by the firewall. Diagnosing the problem then becomes a cross-agency matter to "stitch" together, diagnostics produced by the PSB with diagnostics produced by the agency. This gets very complex, very quickly.

4.3.3 Recommendation

The pros of Option 1 – **message transfer initiated by the service in hop 3 (pull delivery)** - significantly outweigh the cons and this is the recommended approach.

When receiving messages, it is up to the agency to pull them from the PSB.

5 Polled pull delivery versus blocked pull delivery

As mentioned above the concept of pull-delivery can sometimes be mis-interpreted as requiring a non-real-time periodic polling of the PSB. This is not the case. The purpose of this section is to tease out the two main paradigms for implementing pull-delivery. These are:

block-based pull delivery

poll-based pull delivery

5.1 Block-based pull delivery

In a block-based pull delivery model, service code connects to a message queue and "blocks" awaiting the availability of a message on that queue. This blocking mechanism is at the heart of many network and systems programming paradigms. For example:

1. The sockets API [Stevens] to TCP/IP provides a blocked-read facility. This is the basis of such familiar client/server network services as web servers, POP3 servers, ping servers etc. [Stevens]
2. Operating System IO services e.g. The classic blocked read/write functions on Windows/Unix/Mac.
3. The J2EE JMS MessageListener and MessageDrivenBean interfaces provide mechanisms for threads to block-read JMS message queues.
4. Microsoft MSMQ/Biztalk Message APIs.

The blocked pull mechanism has the following desirable attributes:

1. Message handling is very resource friendly. All of the above platforms provide built-in mechanisms to ensure that blocked code does not consume any resources e.g. CPU.
2. Message handling is very timely. Once a message is available, message handling logic can be instantaneously activated to handle the message.
3. Message handling scales horizontally. In each of the platforms mentioned above, mechanisms are available to have multiple "listeners" blocked on a message queue at any one time. This allows message handling throughput to be addressed by adding more listeners without adding complexity and with linear cost growth.

Note: For the above reasons (especially 2 and 3), it is envisaged that blocked-pull delivery will be the transport adapter paradigm of choice for "synchronous" lookup services such as the PSI on the PSB.

5.2 Poll-based pull delivery

In a poll-based pull delivery model, service code periodically connects to the message queue to check for the availability of a message. This method is also referred to as "non-blocking" because it is often implemented using the same APIs as for blocked-pull but switched from the default blocking mode to a non-blocking mode.

Poll-based pull delivery has the advantage that network connections can be established on an as-needed basis. It is appropriate in interactive applications in which message retrieval is performed under human direction. It is also appropriate in batch-processing scenarios in which message handling Terms & Cconditions allow for periodic processing.

Note: For the above reasons, it is envisaged that polled-pull delivery will have some possible business advantages over blocked-pull delivery for "batch" or "asynchronous" services on the PSB.

6 Local push delivery models

As can be seen from section 4, there are as many business reasons as there are technical reasons to prefer pull delivery as the default message delivery model of the PSB.

It does not follow however, that agencies cannot decide themselves to use push-based delivery mechanisms within their own infrastructure. Almost all of the business and technical issues that arise with push delivery cease to be issues when you have end-to-end control over an infrastructure.

Agencies wishing to have push delivery behaviour can configure a machine to pull messages (either blocked or non-blocked) from the PSB and then push them – via whatever means the agency wishes – out into their locally administered applications.

It is anticipated that Reach provides open source software to achieve this "local-push" model. This software will be provided on an "as is" basis and responsibility for operating it lies with the agency concerned.

An overview of the local-push PSB adapter is shown in Figure 4.

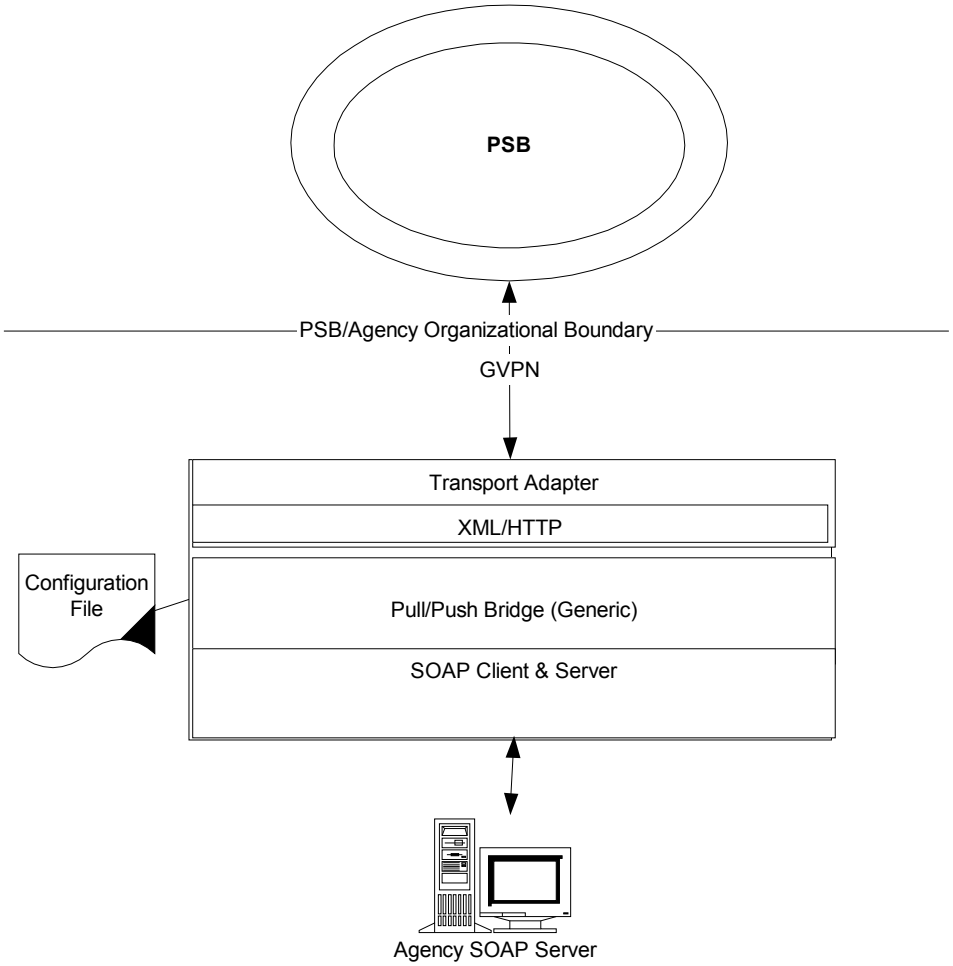

Figure 4 : The local-push PSB adapter

Notes:

The local-push adapter will be configured to XML/HTTP to interact with the PSB core. The local-push adapter can be configured to use polled-read or blocked-read when listening for messages from the PSB.

The local-push adapter will expose a SOAP server for local applications wishing to send messages back to the PSB.

The local-push adapter can be configured to push messages from the PSB to local SOAP servers.

For more information on availability of the local-push adapter, please contact Reach.

7 Examples of industry practice in this area.

7.1 e-grants.gov

The US-based e-grants.gov initiative is similar in many respects to the PSB. It aims to provide a single point of contact for agencies to send and receive XML messages. Two types of interface are provided, as with the PSB, namely a human-facing portal and machine-facing Web Services.

The system is based on an pull delivery model. For details see Agency System to System Integration Toolkit [GRANTS].

7.2 The Swedish e-Government Infrastructure - SHS

The SHS [SHS] system built by the Swedish Agency for Public Management and used widely for inter-agency integration in Sweden, has numerous parallels with the PSB.

It supposed a temporal loose coupling mode. In this mode, agencies push outward bound messages and pull (or "fetch" in SHS terminology) inward bound messages.

7.3 SMTP/POP3 E-mail

The PSB was architected with the success, scalability and robustness of the e-mail model very much in mind.

In e-mail message exchange, the sender (e-mail client) "pushes" outward bound mail (push transmission) and "pulls" inward bound mail (pull delivery), as recommended for the PSB.

The "push" part uses the SMTP protocol. The "pull" part typically uses the POP3 protocol.

Note that users of e-mail packages such as Outlook, Eudora, Thunderbird often have the impression that e-mail is sent (pushed) to their machine. In reality, these applications are pulling (polling) e-mail periodically. The "push" is a convenient, user-level illusion.

Note also that the SMTP protocol allows two SMTP servers to use both PUSH and PULL when talking in Peer To Peer mode with each other - message relay to message relay. In the PSB context, this equates to inter-hub messaging as SMTP servers form a "closed network" of message relays that deliver messages to message queues from where they will be accessed (using pull delivery) by MUAs(Mail User Agents) .

Such a closed network minimizes the organisational responsibility issues. As is common with closed networks (and will be the case with multi-hub PSB) push components reside in DMZs with their own firewall rules etc.

7.4 IMAP4 Webmail

Webmail systems also provide a model suitable for the PSB architecture. With webmail systems, messages remain on message queues maintained by the webmail provider. Users are responsible for pushing e-mail out and for pulling e-mail down using a browser interface. Again, the webmail system does not proactively push e-mails to users, rather it waits for users to connect and pull.

Inter-hub PSB messaging could use a similar arrangement but would necessarily need to be to a different "peer to peer" SLA than the one used for normal agency messaging (client/server).

7.5 RSS/Atom content syndication

When new stories become available, be it on news websites or personal blogs, notifications are sent to subscribers using what are known as "feeds". These feeds are in XML notations – RSS and Atom being two of the most common.

On a surface analysis, content syndication appears to the user to be a push-based system in that you get notified when new stories are available. In reality, this is not how these systems work. As with the SMTP/POP3 e-mail case, the "push" aspect of the technology is a user-level illusion. Behind the scenes, the content is periodically pulled.

8 Business implications of the shared infrastructure required by a PSB push-to-deliver model

Agencies (and indeed all businesses) configure their internet systems to make it easy for internal systems to access the outside world but not the other way around.

Any system that wishes to receive "pushed" information needs to be isolated in what is known as a DMZ (demilitarized zone). This is necessary in order to ensure that untrusted external parties can only access certain parts of the agency systems.

In theory, an agency user, wishing to use a push delivery model to receive messages from the PSB, could set up a box in their DMZ and take complete responsibility and ownership for that box. In reality, the maintenance, troubleshooting and general care and feeding of this box will cause problems. A possible scenario is outlined below:

1. The agency will start out "owning" the PSB integration infrastructure components that are on its premises.
2. Something will go wrong. The agency will understandably contact Reach and say "It is your problem. You fix it."
3. Reach will send an engineer down to the agency (at not insignificant cost) and find that some hardware/software combination has malfunctioned.
4. There will be a disagreement over who pays depending on the nature of the malfunction.
5. After a while, Reach will have no option from a business perspective but to say "This is a Reach box. Reach owns it. Reach will set it up and manage it. The agency must agree not to touch it."

This gradual evolution from no-shared infrastructure (other than the GVPN) to effectively having to establish "PSB embassies" within agencies will carry with it significant cost implications for Reach.

These issues can be avoided by eschewing PSB push delivery in favour of pull delivery.

In scenarios which, for whatever reason, absolutely require push delivery, co-locating the service with the PSB may be the best way of dealing with these issues.

9 Summary and recommendations

Both the push delivery and pull delivery models have their place. Indeed the TDS asks for both (see Appendix 1).

For the reasons outlined in this paper, Reach advises:

1. That pull delivery be the default message delivery mechanism of the PSB.
2. That pull delivery be supported by all transport adapters in line with the "all adapters are equal" principle of the TDS.
3. That agencies be fully appraised of the business and technical disadvantages of a pull-delivery model.
4. That agencies who absolutely require a push-delivery model either:
    1. Utilise a hybrid push/pull mechanism (see section 6)
    2. Co-locate their service with the PSB
6. That vendors of technologies likely to feature in PSB Transport Adapters be appraised of the architecture and invited to provide

configurations of their systems that work well with the “pull delivery” model.

10 Appendix 1 - Relevant Sections from the TDS

10.1 From section 3.2.1 of TDS 1.1 – The Loose Coupling Principle.

> "Functionality at the edges of the PSB will communicate in a loosely coupled fashion. Reach takes loose coupling (temporal, spatial, contextual) very seriously and achieves it via basing component interactions on asynchronous data-centric interfaces".

10.2 From section 3.2.6 of TDS 1.1 – The Agency Footprint Principle.

> It must be possible for agencies to use the PSB without a significant PSB footprint. Ideally, it must be possible to interact with the PSB using nothing more than browser technology (this is considered to be zero footprint – human interface) and HTTP/XML (this is considered to be zero footprint – process interface). Higher footprints should be at the discretion of the agencies themselves.

10.3 From View 1 – Service Oriented Architecture View

> "
> - The term "loose coupling" will be applied rigorously to PSB Enabled Web Services incorporating the following facets:
>   - Temporal decoupling – services are temporally decoupled. That is to say, inter-service communication is asynchronous. Synchronous facilities are provided on top of the asynchronous communication substrate. See Synchronous/Asynchronous View
>
> "

10.4 From PSB Phase 1 Procurement Advisory Note 08

"An important principle for the PSB is that all adaptors are equal and provide access to dumb reliable message delivery functionality. In particular, if agencies are using the same messaging system as the PSB system, this basic functionality is all that should be provided via this service. There should be no creep of 'value adding' functionality that is specific to this channel which would inevitably result in lock-in to the chosen messaging product. "

11 References

[TDS] Http://www.reach.ie/tds

[GRANTS] http://www.grants.gov/assets/Agency_Integration_Toolkit.doc

also

http://www.grants.gov/DoingBusinessIT

[Helland] Autonomous Computing: Fiefdoms and Emissaries

http://whitepapers.zdnet.co.uk/0,39025945,60045295p-39000483q,00.htm

[SHS] http://www.statskontoret.se/shs/pdf/shs-architecture.pdf

[Stevens] TCP/IP Illustrated, Volumes 1-3, W. Richard Stevens, Addisson Wesley (in particular volume 3)

12 Circulation