



---

## **E-Procurement Implementation Architecture**

Version 0.4 - 12/03/2007

Preliminary Release

---

**National E-Health Transition Authority Ltd**

Level 25  
56 Pitt Street  
Sydney, NSW, 2000  
Australia.  
[www.nehta.gov.au](http://www.nehta.gov.au)

**Disclaimer**

NEHTA makes the information and other material ("Information") in this document available in good faith but without any representation or warranty as to its accuracy or completeness. NEHTA cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

**Document Control**

This document is maintained in electronic form. The current revision of this document is located on the NEHTA Web site and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is of the latest revision.

**Copyright © 2007, NEHTA.**

This document contains information which is protected by copyright. All Rights Reserved. No part of this work may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without the permission of NEHTA. All copies of this document must include the copyright and other information contained on this page.

# Table of contents

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
1.1	Purpose .....	2
1.2	Document Context .....	2
1.3	Intended Audience .....	2
1.4	Background.....	3
1.5	Feedback .....	3
1.6	Structure of this document.....	3
<b>2</b>	<b>Implementation Overview .....</b>	<b>4</b>
2.1	Naming Conventions .....	4
2.2	Sending Documents from Buyer to Supplier .....	4
2.2.1	Push Model.....	4
2.2.2	Pull Model .....	5
2.3	Sending Documents from Hubs or Buyers to Suppliers.....	5
2.3.1	Push Model.....	5
2.3.2	Pull Model .....	6
<b>3</b>	<b>Service Definitions.....</b>	<b>8</b>
3.1	Common Type Definitions .....	8
3.2	BuyerPushPortType .....	8
3.3	SupplierPushPortType.....	9
3.4	SupplierPullPortType .....	10
<b>4</b>	<b>Example Interactions.....</b>	<b>17</b>
4.1	Supplier Context .....	17
4.2	Direct Connect.....	17
4.3	Buyer Push Interface supported by Hub .....	18
4.4	Buyer Push Interface Supported by Supplier.....	19
4.5	Supplier Pull Interface Supported by Hub.....	20

This page has been left blank intentionally.

# Executive Summary

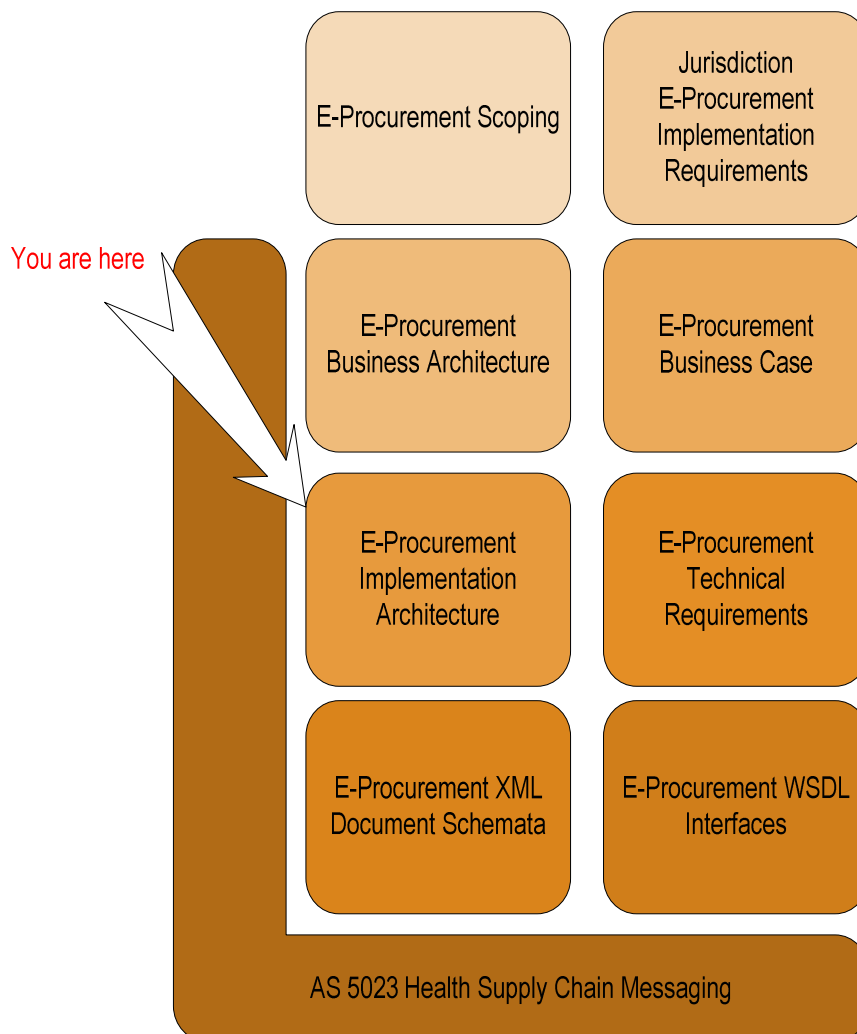
This implementation architecture explains the paradigm of interactions between the three roles in E-Procurement: Buyers, Hubs and Suppliers. It then goes on to document the Web Services interfaces that are used to facilitate the exchange of business documents. Finally, some example interactions are shown using UML Interaction Diagrams.

# 1 Introduction

## 1.1 Purpose

This document explains the detail of how to implement the NEHTA E-Procurement Business Architecture. It contains explanations of the attached XML Schema Specifications, and WSDL file, as well as the other Web Services infrastructure required to implement E-Procurement Messaging. It provides a set of conformance statements and a testing methodology that will be used by the NEHTA E-Procurement Testing Body to certify that Hub Service Providers meet this specification. This complements the conformance statements and testing methodology given in the E-Procurement Technical Requirements document, which should be read in concert with this Implementation Architecture.

## 1.2 Document Context



## 1.3 Intended Audience

This document is intended for all parties that will implement the NEHTA Standard E-Procurement Messaging Architecture:

- State, Territory and Commonwealth Health Department procurement divisions,
- Hospitals and Area Health Services,
- E-Procurement Hub service providers,
- Suppliers who wish to send and receive messages directly with Health buyers without using a Hub service, and
- The NEHTA E-Procurement Testing Body.

## 1.4 Background

The Informational perspective of this Architecture is given in a non-format specific form by the Australian Standard from Health Supply Chain Messaging, AS 5023. XML Schemata chosen for the technical perspective, and to be used as the NEHTA E-Procurement technical document standard have been chosen through a thorough evaluation process, documented in an evaluation paper entitled *Business Document Format Choices for Health E-Procurement*. This paper is available upon request.

The Web Services standards chosen for use in this Implementation Architecture follow the NEHTA Secure Messaging Architecture, which can be downloaded from the NEHTA web site.

The organisational roles and process framework within which E-Procurement messaging is to be conducted is described in the E-Procurement Business Architecture.

## 1.5 Feedback

This is a preliminary release version of this document. Comments can be submitted via the NEHTA website.

## 1.6 Structure of this document

Section 2 of this document is the Implementation Overview. It provides naming conventions for data types used in the WSDL defining operations for the transfer of e-procurement documents. It then describes the possible paradigms of interaction to achieve the business goal of transferring procurement documents between buyers and their suppliers, using hubs where necessary.

Section 3 contains the Service Definitions of the Web Services that are defined in order to transfer XML representations of procurement documents. Each subsection describes a Port Type and the operations that the Port Type supports.

Section 4 shows some Example Interactions between Buyers, Suppliers and Hubs that illustrate how the Web Services defined in Section 3 should be used to achieve the desired business communications.

## 2 Implementation Overview

This section describes the technical message passing scenarios that facilitate E-Procurement documents getting from Buyers to Suppliers and vice versa.

### 2.1 Naming Conventions

There are two main roles in the interchange of e-procurement documents: Buyer and Supplier. The third major role is the Hub, but the hub always acts as a proxy buyer to a supplier, and as a proxy supplier to a buyer. It implements and uses the same interfaces as buyers and suppliers.

WSDL port types are named in the following way:

```
<InvokerRole>[Push|Pull]PortType
```

For example a port type designed for invocation by a Buyer (or a hub acting on behalf of a buyer), using the push model will be called `BuyerPushPortType`, and a port type designed for invocation by a supplier (or a hub acting as a supplier), using the pull model will be called `SupplierPullPortType`.

WSDL operations within a port type are always named:

```
[Push|Pull]<DocumentName>
```

For example, the operation to push an Invoice in the `BuyerPushPortType` will be called `PushInvoice`, and an operation to pull a Purchase Order in the `SupplierPullPortType` will be called `PullPurchaseOrder`.

The input and output message types for operations are named according to Web Services conventions. For example the input message type for `PullPurchaseOrder` is named `PullPurchaseOrderInMsg` and its output message is named `PullPurchaseOrderOutMsg`.

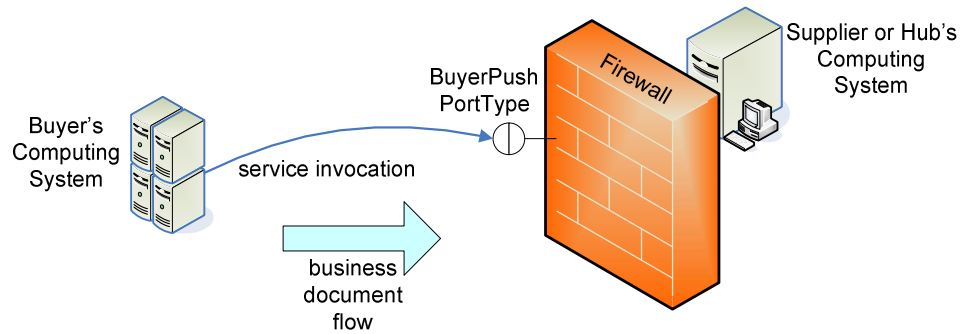
Please refer to the NEHTA Guidelines for Implementing Web Services [GIWS] for details on Web Services conventions to be used in implementing this standard.

### 2.2 Sending Documents from Buyer to Supplier

#### 2.2.1 Push Model

The most common means for a Buyer to send a business document to a Supplier is to invoke the common "push style" web service interface that is supported by Hubs and direct-connect Suppliers alike. The `SupplierPush<version>.wsdl` file contains a port type called `SupplierPushPortType` which will be able to accept a document of one of the types that Suppliers send to their Buyers (i.e. Purchase Order, or Purchase Order Change) using a Push operation.

In addition, the port type has a generic `PushDocument` operation to allow buyers to send documents to suppliers that are not nominated in the NEHTA E-Procurement Architecture. **NOTE:** It is expected that the generic operation will *not* be used to transmit any of the documents for which specific operations have been designed. See Figure 1 for an illustration of this scenario.



**Figure 1: Supplier invokes "push style" interface of Web Service**

### 2.2.2 Pull Model

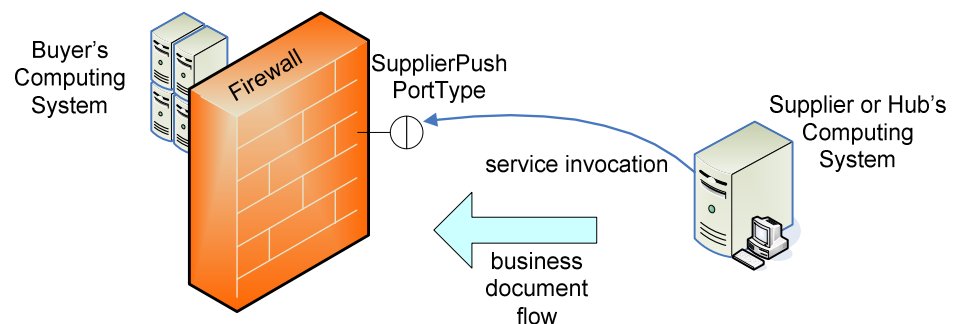
Buyers (i.e. the jurisdictions) will not support an interaction style that requires them to poll for new documents by invoking "pull style" interfaces. Also, hubs that act on behalf of buyers will expect that supplier sending them documents will make invocations at web services ports implemented by the hub, rather than have the hub pull documents from the Supplier. Therefore there is no `BuyerPullPortType` defined. The pull model will be supported in the other direction, however, as not all suppliers will have the technical sophistication to implement web services and make them available through their firewalls. See Section 2.3.2 for details.

## 2.3 Sending Documents from Hubs or Buyers to Suppliers

### 2.3.1 Push Model

The most common means for a direct-connect Supplier, or Hub, to send a business document to a Buyer is to invoke a "push style" web services interface. The `BuyerPush.wsdl` file contains a port type called `BuyerPushPortType` which has operations defined to accept documents of the types that Buyers expect (i.e. Purchase Order Response, Despatch Advice or Invoice).

In addition, it has a generic `PushDocument` operation to allow buyers to send documents to suppliers that are not nominated in the NEHTA E-Procurement Architecture. **NOTE:** It is expected that the generic operation will *not* be used to transmit any of the documents for which specific operations have been designed.



**Figure 2: Buyer invokes "push style" interface of Web Service**

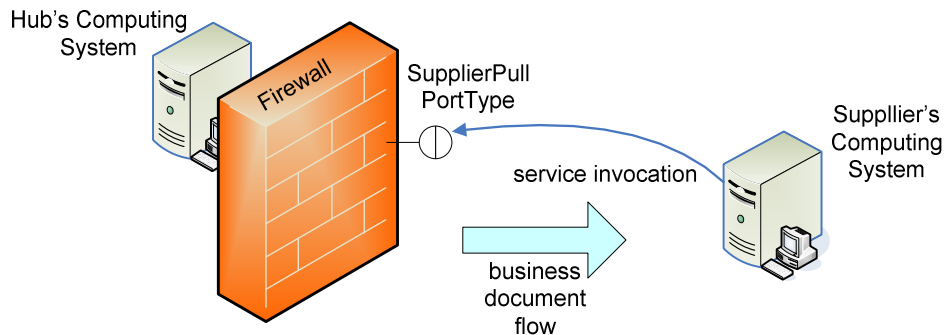
### 2.3.2 Pull Model

Hubs will allow their supplier clients to poll for new documents rather than supporting a web service interface through their firewall. To facilitate this the `SupplierPull.wsdl` file contains a port type called `SupplierPullPortType`. It contains an operation called `GetQueueStatus` which returns information to the supplier about how many documents of various types are queued for it. The port also has pull operations which accept requests for a document of one of the types that Suppliers expect (Purchase Order, Purchase Order Change and Purchase Order Cancel).

In addition the `GetQueueStatus` operation will return a list of queues for documents other than these nominated types, and the port also has a generic operation which allows for documents in these queues to be retrieved by the supplier. **NOTE:** The pull operations for the specific nominated documents must be used to retrieve these documents, and the generic operation may only be used for documents other than those listed in the NEHTA E-Procurement Architecture.

The Supplier Pull interface achieves the same task as the Buyer Push interface; i.e. to move documents from the buyer (via its hub) to the supplier. Compare the business document flow direction of the pull model in Figure 3 to that of the equivalent push model in Figure 1.

The polling interface is less efficient, as it requires the `GetQueueStatus` operation to be called at regular intervals to check whether any documents are waiting. However it means that a supplier need not implement a web service or perform any manipulation of its firewall to permit invocations to be made, which require a level of technical sophistication higher than that currently available in many supplier companies.

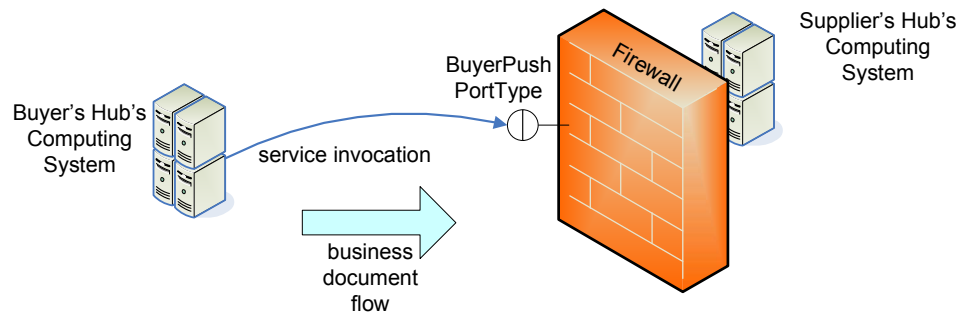


**Figure 3: Supplier invokes "pull style" interface of Hub's Web Service**

## 2.4 Hub Interconnect

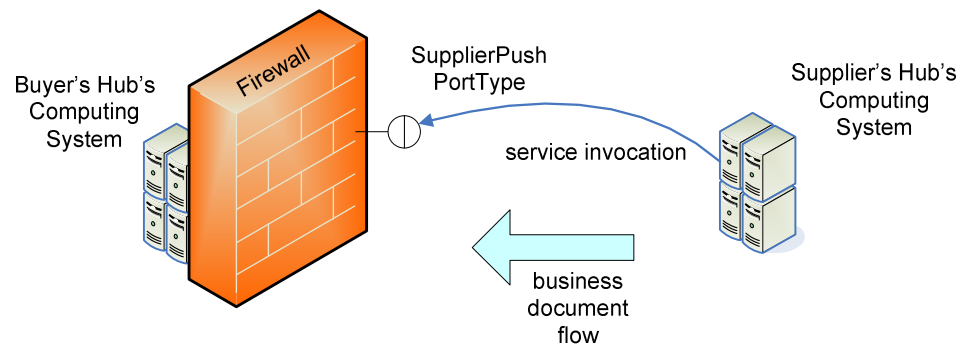
The scenarios above all involve a direct connection between buyer and supplier, or the use of a hub which stores and forwards business documents between these two parties. The E-Procurement Business Architecture allows for the case where a buyer and a supplier do not connect to the same hub. This means that an additional step is required: the transfer of the document from the hub that is connected to the buyer to the hub that is connected to the suppliers, or vice versa. This is known as *Hub Interconnect*.

Unless hubs already have a mutually agreeable mechanism for interconnect that does not use this standard, then the following invocations are required. A hub that has business documents from a buyer that it received via its support of the `BuyerPushPortType` must pass these on to the hub connected to the supplier to whom the document is addressed, using the supplier's hub's `BuyerPushPortType`. This is shown in Figure 4.



**Figure 4: Buyer's Hub interconnects to Supplier's Hub**

Conversely, a hub that has received messages from a supplier destined for a buyer connected to another hub, via a `SupplierPushPortType` must forward these messages to the buyer's hub using its `SupplierPushPortType`. This is shown in Figure 5.



**Figure 5: Supplier's Hub interconnects to Buyer's Hub**

## 3 Service Definitions

This section documents the major XML argument types and Port Types defined in the WSDL files: BuyerPush\_v0\_5.wsdl, SupplierPush\_v0\_6.wsdl and SupplierPull\_v0\_7.wsdl.

### 3.1 Common Type Definitions

#### Type Definitions

Other than the UBL document types, there is only one XML type definition that is used by all port types:

```
<xsd:complexType name="Nil">
    <xsd:sequence/>
</xsd:complexType>
```

The Nil type is used as the return message when documents are pushed, and as an outgoing message when documents are being pulled. A web services invocation of a Push operation that correctly returns a Nil without any protocol level exceptions being raised is assumed to have correctly delivered its document.

### 3.2 BuyerPushPortType

#### Operation: PushPurchaseOrder

##### Target Namespace

`http://nehta.gov.au/SupplyChain/BuyerPush`

##### Preconditions

None.

##### Input

`name="PushPurchaseOrder" type="ublorder:Order"`

##### Output

`name="PushPurchaseOrderResponse" type="tns:nil"`

##### Behaviour

PushPurchaseOrder transmits a UBL Order document from a buyer to a supplier or hub, returning a Nil upon successful completion.

#### Operation: PushPurchaseOrderCancel

##### Target Namespace

`http://nehta.gov.au/SupplyChain/BuyerPush`

##### Preconditions

None.

##### Input

`name="PushPurchaseOrderCancel"`  
`type="ublordercancellation:OrderCancellation"`

**Output**

```
name="PushPurchaseOrderCancelResponse"  
type="tns:Nil"
```

**Behaviour**

PushPurchaseOrderCancel transmits a UBL OrderCancellation document from a buyer to a supplier or hub, returning a Nil upon successful completion.

**Operation: PushPurchaseOrderChange****Target Namespace**

```
http://nehta.gov.au/SupplyChain/BuyerPush
```

**Preconditions**

None.

**Input**

```
name="PushPurchaseOrderChange"  
type="ublorderchange:OrderChange"
```

**Output**

```
name="PushPurchaseOrderChangeResponse"  
type="tns:Nil"
```

**Behaviour**

PushPurchaseOrderChange transmits a UBL OrderChange document from a buyer to a supplier or hub, returning a Nil upon successful completion.

### 3.3 SupplierPushPortType

**Operation: PushPurchaseOrderResponse****Target Namespace**

```
http://nehta.gov.au/SupplyChain/SupplierPush
```

**Preconditions**

None.

**Input**

```
name="PushPurchaseOrderResponse"  
type="ubl:orderresponse:OrderResponse"
```

**Output**

```
name="PushPurchaseOrderResponseResponse"  
type="tns:Nil"
```

**Behaviour**

PushPurchaseOrderResponse transmits a UBL OrderResponse document from a buyer to a supplier or hub, returning a Nil upon successful completion.

**Operation: PushDespatchAdvice****Target Namespace**

http://nehta.gov.au/SupplyChain/SupplierPush

**Preconditions**

None.

**Input**

name="PushDespatchAdvice"  
type="ubl:despatchadvice:DespatchAdvice"

**Output**

name="PushDespatchAdviceResponse" type="tns:Nil"

**Behaviour**

PushDespatchAdvice transmits a UBL DespatchAdvice document from a buyer to a supplier or hub, returning a Nil upon successful completion.

**Operation: PushInvoice****Target Namespace**

http://nehta.gov.au/SupplyChain/SupplierPush

**Preconditions**

None.

**Input**

name="PushInvoice" type="ubl:invoice:Invoice"

**Output**

name="PushInvoiceResponse" type="tns:Nil"

**Behaviour**

PushInvoice transmits a UBL Invoice document from a buyer to a supplier or hub, returning a Nil upon successful completion.

**3.4 SupplierPullPortType****Data Type: QueueStatus****Target Namespace**

http://nehta.gov.au/SupplyChain/SupplierPull

**Definition**

```
<xsd:complexType name="QueueStatus">
  <xsd:sequence>
    <xsd:element name="OrderQueueSize" minOccurs="1"
      maxOccurs="1" type="xsd:integer"/>
    <xsd:element name="OrderCancelQueuesize"
      minOccurs="1" maxOccurs="1"
      type="xsd:integer"/>
  </xsd:sequence>
</xsd:complexType>
```

```

<xsd:element name="OrderChangeQueueSize"
  minOccurs="1" maxOccurs="1"
  type="xsd:integer"/>
<xsd:element name="OtherQueues"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Name" minOccurs="1"
        maxOccurs="1" type="xsd:string"/>
      <xsd:element name="Size" minOccurs="1"
        maxOccurs="1" type="xsd:integer"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>

```

**Purpose**

The QueueStatus type is the return type for the GetQueueStatus operation. It represents the current length of the queues for documents that Suppliers would pull from a hub (Purchase Orders, Purchase Order Cancellations and Purchase Order Changes). In addition it has the ability to inform a supplier that additional documents are queued in named queues, using a set of OtherQueues elements.

**Data Type: OrderQueueHead****Target Namespace**

<http://nehta.gov.au/SupplyChain/SupplierPull>

**Definition**

```

<xsd:complexType name="OrderQueueHead">
  <xsd:sequence>
    <xsd:element name="RemQueueSize" minOccurs="1"
      maxOccurs="1" type="xsd:integer"/>
    <xsd:element name="QueueHead" minOccurs="0"
      maxOccurs="1" type="ublorder:Order"/>
  </xsd:sequence>
</xsd:complexType>

```

**Purpose**

The OrderQueueHead type is the return type for the PullPurchaseOrder operation. It contains two elements: RemQueueSize, which indicates how many documents remain on the Order

queue at the hub, and QueueHead which is the Order document that has just been popped from the head of the queue.

## Data Type: OrderCancelQueueHead

### Target Namespace

`http://nehta.gov.au/SupplyChain/SupplierPull`

### Definition

```
<xsd:complexType name="OrderCancelQueueHead">
  <xsd:sequence>
    <xsd:element name="RemQueueSize" minOccurs="1"
      maxOccurs="1" type="xsd:integer"/>
    <xsd:element name="QueueHead" minOccurs="0"
      maxOccurs="1"
      type="
        "ublordercancellation:OrderCancellation"/>
  </xsd:sequence>
</xsd:complexType>
```

### Purpose

The OrderCancelQueueHead type is the return type for the PullPurchaseOrderCancel operation. It contains two elements: RemQueueSize, which indicates how many documents remain on the Order queue at the hub, and QueueHead which is the Order Cancellation document that has just been popped from the head of the queue.

## Data Type: OrderChangeQueueHead

### Target Namespace

`http://nehta.gov.au/SupplyChain/SupplierPull`

### Definition

```
<xsd:complexType name="OrderChangeQueueHead">
  <xsd:sequence>
    <xsd:element name="RemQueueSize" minOccurs="1"
      maxOccurs="1" type="xsd:integer"/>
    <xsd:element name="QueueHead" minOccurs="0"
      maxOccurs="1"
      type="ublorderchange:OrderChange"/>
  </xsd:sequence>
</xsd:complexType>
```

### Purpose

The OrderChangeQueueHead type is the return type for the PullPurchaseOrderChange operation. It contains two elements:

RemQueueSize, which indicates how many documents remain on the Order queue at the hub, and QueueHead which is the Order Change document that has just been popped from the head of the queue.

### **Data Type: QueueName**

#### **Target Namespace**

`http://nehta.gov.au/SupplyChain/SupplierPull`

#### **Definition**

```
<xsd:complexType name="QueueName">
  <xsd:sequence>
    <xsd:element name="QueueName" minOccurs="1"
      type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

#### **Purpose**

The QueueName type contains a single string valued element which is used as the input to the generic PullDocument operation to indicate which queue a document should be pulled from.

### **Data Type: OtherQueueHead**

#### **Target Namespace**

`http://nehta.gov.au/SupplyChain/SupplierPull`

#### **Definition**

```
<xsd:complexType name="OtherQueueHead">
  <xsd:sequence>
    <xsd:element name="RemQueueSize" minOccurs="1"
      type="xsd:integer"/>
    <xsd:element name="QueueHead" minOccurs="0"
      type="xsd:anyType"/>
  </xsd:sequence>
</xsd:complexType>
```

#### **Purpose**

The OtherQueueHead type is the return type for the generic PullDocument operation. It contains a RemQueueSize element which is an integer indicating how many documents remain in the named queue, and a QueueHead element which is of type anyType for the document that has just been popped off the named queue.

### **Operation: GetQueueStatus**

#### **Target Namespace**

`http://nehta.gov.au/SupplyChain/SupplierPull`

**Preconditions**

None.

**Input**

name="GetQueueStatus" type="tns:Nil"

**Output**

name="GetQueueStatusResponse"  
type="tns:QueueStatus"

**Behaviour**

The GetQueueStatus operation takes a Nil input message, and returns a QueueStatus type, containing the lengths of all of the queues of documents awaiting the invoking supplier.

**Operation: PullPurchaseOrder****Target Namespace**

<http://nehta.gov.au/SupplyChain/SupplierPull>

**Preconditions**

In order for a Purchase Order to be successfully pulled the caller should first determine if there are any queued by invoking the GetQueueStatus operation. This call will not fail if there are no Purchase Order documents queued, but the invocation will return no document.

**Input**

name="PullPurchaseOrder" type="tns:Nil"

**Output**

name="PullPurchaseOrderResponse"  
type="tns:OrderQueueHead"

**Behaviour**

PullPurchaseOrder has a Nil typed input. It returns an OrderQueueHead element, which contains the Purchase Order at the head of the queue for the invoking Supplier, if the queue is non-empty. The other component of the OrderQueueHead is the remaining queue size, indicating how many more Purchase Orders are still to be pulled by repeated invocations of this operation.

**Operation: PullPurchaseOrderCancel****Target Namespace**

<http://nehta.gov.au/SupplyChain/SupplierPull>

**Preconditions**

In order for a Purchase Order Cancellation to be successfully pulled the caller should first determine if there are any queued by invoking the GetQueueStatus operation. This call will not fail if there are no Purchase Order Cancellation documents queued, but the invocation will return no document.

**Input**

name="PullPurchaseOrderCancel" type="tns:Nil"

**Output**

```
name="PullPurchaseOrderCancelResponse"  
type="tns:OrderCancelQueueHead"
```

**Behaviour**

PullPurchaseOrderCancel has a Nil typed input. It returns an OrderCancelQueueHead element, which contains the Purchase Order Cancellation at the head of the queue for the invoking Supplier, if the queue is non-empty. The other component of the OrderCancelQueueHead is the remaining queue size, indicating how many more Purchase Order Cancellations are still to be pulled by repeated invocations of this operation.

**Operation: PullPurchaseOrderChange****Target Namespace**

```
http://nehta.gov.au/SupplyChain/SupplierPull
```

**Preconditions**

In order for a Purchase Order Change to be successfully pulled the caller should first determine if there are any queued by invoking the GetQueueStatus operation. This call will not fail if there are no Purchase Order Change documents queued, but the invocation will return no document.

**Input**

```
name="PullPurchaseOrderChange" type="tns:Nil"
```

**Output**

```
name="PullPurchaseOrderChangeResponse"  
type="tns:OrderChangeQueueHead"
```

**Behaviour**

PullPurchaseOrderChange has a Nil typed input. It returns an OrderChangeQueueHead element, which contains the Purchase Order Cancellation at the head of the queue for the invoking Supplier, if the queue is non-empty. The other component of the OrderChangeQueueHead is the remaining queue size, indicating how many more Purchase Order Changes are still to be pulled by repeated invocations of this operation.

**Operation: PullDocument****Target Namespace**

```
http://nehta.gov.au/SupplyChain/SupplierPull
```

**Preconditions**

In order for a document to be successfully pulled from a named queue, the caller should first determine if there are any documents queued by invoking the GetQueueStatus operation. This call will not fail if there are no documents queued in the named queue, but the invocation will return no document.

**Input**

```
name="PullDocument" type="tns:QueueName"
```

**Output**

```
name="PullDocumentResponse"  
type="tns:OtherQueueHead"
```

**Behaviour**

PullDocument has a QueueName typed input containing a string which names the queue from which a document should be pulled. It returns an OtherQueueHead element, which contains a document for the invoking Supplier within an anyType element from the head of the queue nominated by the QueueName input, if the queue is non-empty. The other component of the OtherQueueHead is the remaining queue size, indicating how many more documents are still to be pulled by repeated invocations of this operation with the same queue name as input.

## 4 Example Interactions

This section provides UML Interaction diagrams demonstrating the use of the Web Services port types defined in Section 3.

### 4.1 Supplier Context

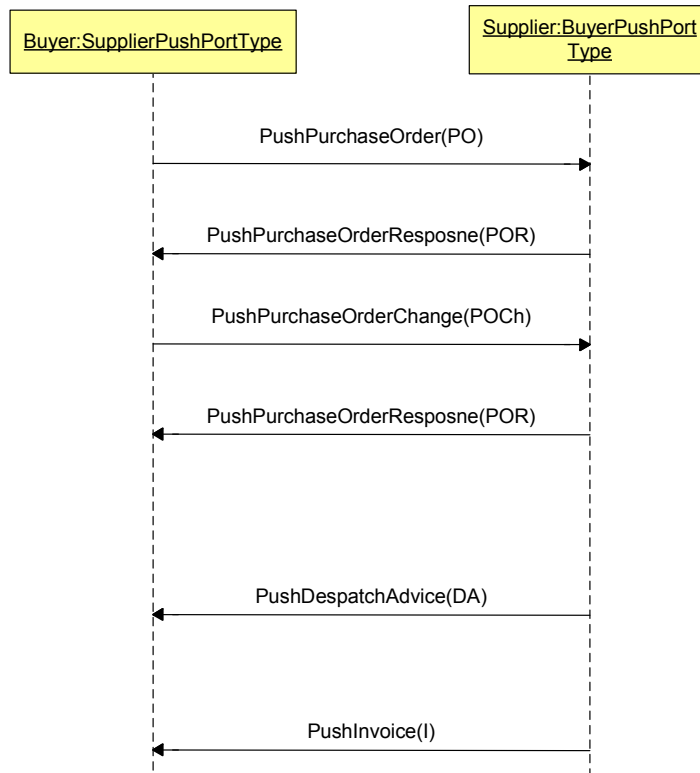
Hubs will usually deliver business documents to suppliers in a form other than the NEHTA standard, using protocols other than Web Services interactions. This is due to the fact that most suppliers already engage in e-procurement with buyers in other industry sectors, using document types and protocols specified for those industries. Hubs provide message transformation services that allow NEHTA standard messages to be translated into other document types. It is obviously easier for a supplier to accept business documents from all its customers via the same hub, using the same format as it already accepts from other customers.

However, there are some suppliers who do not yet implement any e-procurement, and have jurisdictional health buyers as their major customers. They may choose to implement e-procurement using this standard.

### 4.2 Direct Connect

It is recommended that direct connections between Health Jurisdiction buyers and their suppliers are only attempted in cases where suppliers have sufficient technical sophistication to implement Web Services interfaces and make these implementations available through their firewalls. Although a pull-style supplier interface is defined in this specification, this is intended for cases in which hubs wish to make a NEHTA Standard Web Service available for their less technically capable suppliers to poll. Therefore this document leaves discussion of the use of a combination of push and pull style interactions to Section 4.5, which is about Hubs supporting Pull interfaces.

In this section we assume that direct connection is implemented by a Supplier supporting a Web Service with Port Type `BuyerPushPortType` and a Buyer supporting a Web Service of Port Type `SupplierPushPortType`.



**Figure 6: Direct Connect using Push model**

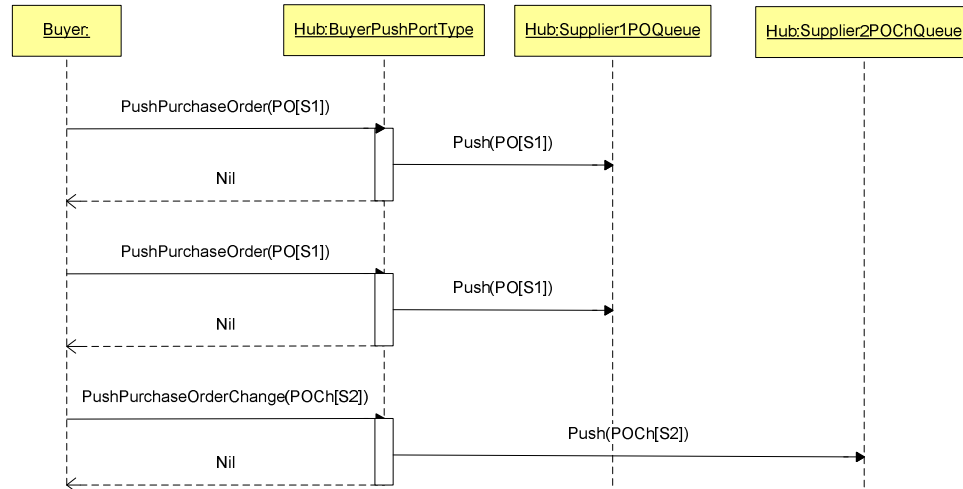
The Interaction Diagram in Figure 6 shows an example of a buyer and a supplier interacting by pushing documents to one another. Each of the trading partners will make a port type of the form `<Invoker>PushPortType` available through their firewalls, and the other partner will invoke the `Push<Document>` operations available through that port. The example above is one of the possible instances of the Business Process specified in the E-Procurement Business Architecture [EPROC-BUS].

### 4.3 Buyer Push Interface supported by Hub

The role of a hub is to store and forward business documents for both buyers and their suppliers. In Figure 7 we show the invocation of two different push-style operations supported by the hub's implementation of the `BuyerPushPortType`. The supplier who is the addressee of a business document is indicated by placing a code in square brackets in the argument representation. In this example, "S1" indicates that a document is addressed to Supplier1, and "S2" indicates that the document is addressed to Supplier2. We assume some sort of per-supplier/per-document-type queue is available for queuing documents to be sent to Suppliers. This representation shows a simplified scenario where documents are to be retransmitted without any transformations or other manipulations.

The Interaction diagram below shows a single buyer "Buyer" making three invocations on an web service at the hub implementing a

BuyerPushPortType port. In each case a successful return of the invocation with a dummy Nil type argument indicates that the hub has successfully received the document, and has it enqueued for collection by its addressee. (See Figure 8 and Figure 9 for alternative push- and pull-style interactions to deliver these documents to suppliers).



**Figure 7: Buyer invokes Push interface at Hub**

## 4.4 Buyer Push Interface Supported by Supplier

The example shown in Figure 8 is the delivery of the messages that were queued at the Hub as a result of the interactions with a single buyer as shown in Figure 7. (The alternative pull-style delivery is shown in Figure 9.)

Figure 8 shows a simplistic representation of what happens inside the hub: a queue is maintained of each document type for each supplier that the hub serves. The interaction proceeds by having the hub pop the first document off the first queue, and deliver it to its addressee by making an invocation of the appropriate operation defined in the BuyerPushPortType. It then proceeds to pop documents from the first queue until it receives a nil return value, indicating that the queue is empty. The hub then proceeds to pop documents from the head of the next queue, and deliver them by invoking Web Services operations on the port at the supplier addressee. In this example there have been two Purchase Orders placed in the queue for Supplier1, and a single Purchase Order Change placed in the queue for Supplier2. These are delivered by invoking the PushPurchaseOrder and PushPurchaseOrderChange operations, respectively.

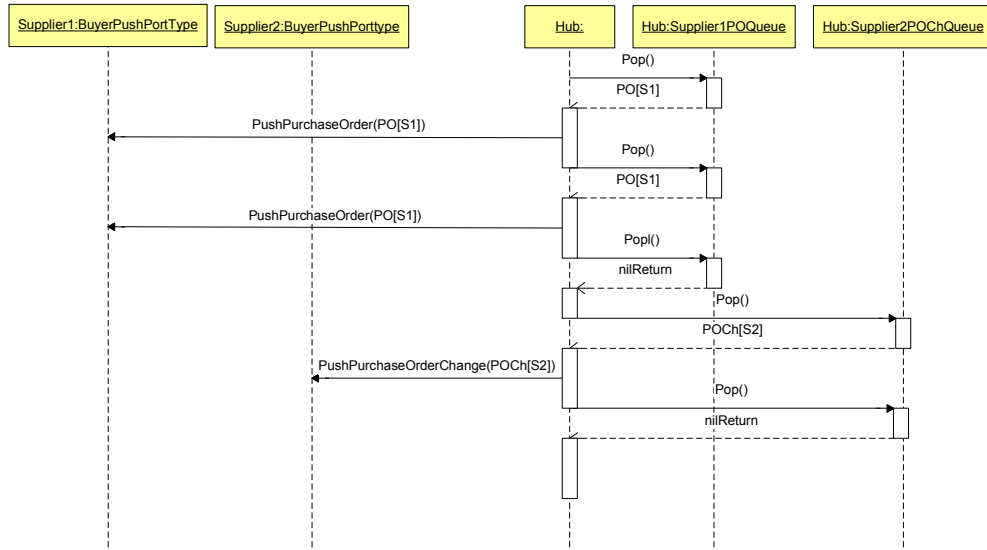


Figure 8: Hub invokes Push interface at Supplier

## 4.5 Supplier Pull Interface Supported by Hub

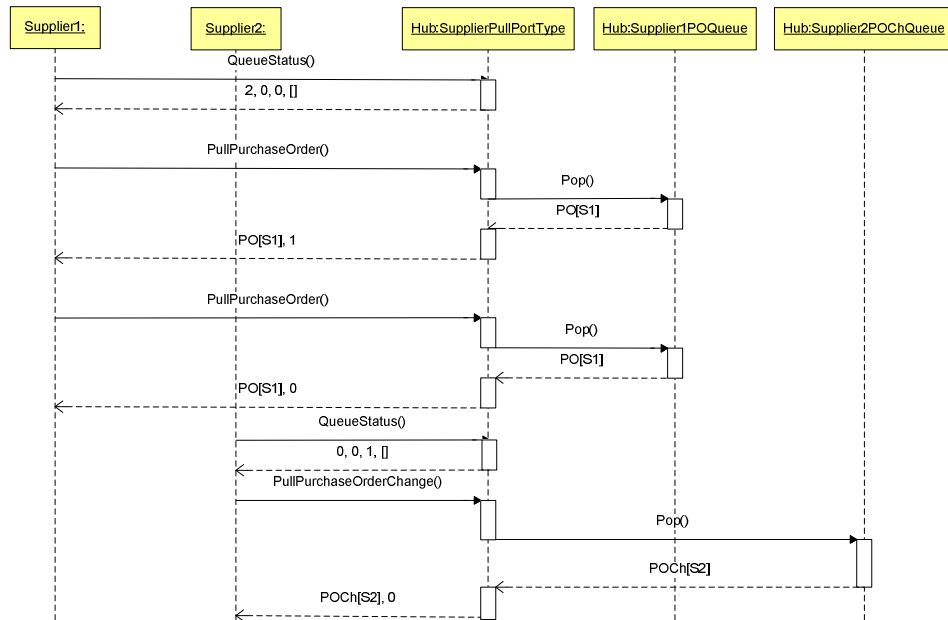


Figure 9: Supplier invoked Pull interface at Hub

The example shown in Figure 9 is the delivery of the messages that were queued at the Hub as a result of the interactions with a single buyer as shown in Figure 7. The pull model of interaction between suppliers and hubs is facilitated by the `SupplierPullPortType`. This is a polling interface

supported by hubs that allows suppliers to check whether there are any new documents queued for them, using the `QueueStatus` operation, and to retrieve documents using a range of "Pull" operations. This operation returns the length of the queues for all the document types that suppliers receive: Purchase Orders, Purchase Order Cancellations, and Purchase Order Changes. The `QueueStatus` operation also returns a list of other queues, and their current sizes, for documents not included in the NEHTA standard. Once a supplier is aware of the status of the various document queues at the hub, it can invoke the appropriate operations to retrieve the documents. Only one document is delivered per invocation, and an additional return parameter indicates how many documents remain in the queue. In the example in Figure 9, Supplier1 has two Purchase Orders waiting for it, and it invokes the `PullPurchaseOrder` operation twice to retrieve them. Supplier2 has a single Purchase Order Change document waiting for it, which it discovers using the `QueueStatus` operation. It retrieves this document by invoking `PullPurchaseOrderChange`.