Digitally signed by National E-Health Transition Authority Ltd
Date: 2012.12.14 13:55:50 +11'00'

nehta

E-Procurement Technical Architecture

Supply Chain

Version 1.1 - 13/02/2008

Final

National E-Health Transition Authority

National E-Health Transition Authority Ltd
Level 25
56 Pitt Street
Sydney, NSW, 2000
Australia.
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 © 2008, 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 without the permission of NEHTA. All copies of this document must include the copyright and other information contained on this page.

Table of Contents

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. Some example interactions are shown using UML Interaction Diagrams. It then goes on to document the XML types and Web Services interfaces that are used to facilitate the exchange of business documents.

1 Introduction

1.1 Purpose

This document explains the detail of how to implement the NEHTA E-Procurement Business Architecture [EPBA2007]. It contains explanations of the attached XML Schema Specifications, and WSDL files, as well as the other Web Services infrastructure required to implement E-Procurement Messaging.

1.2 Document Context

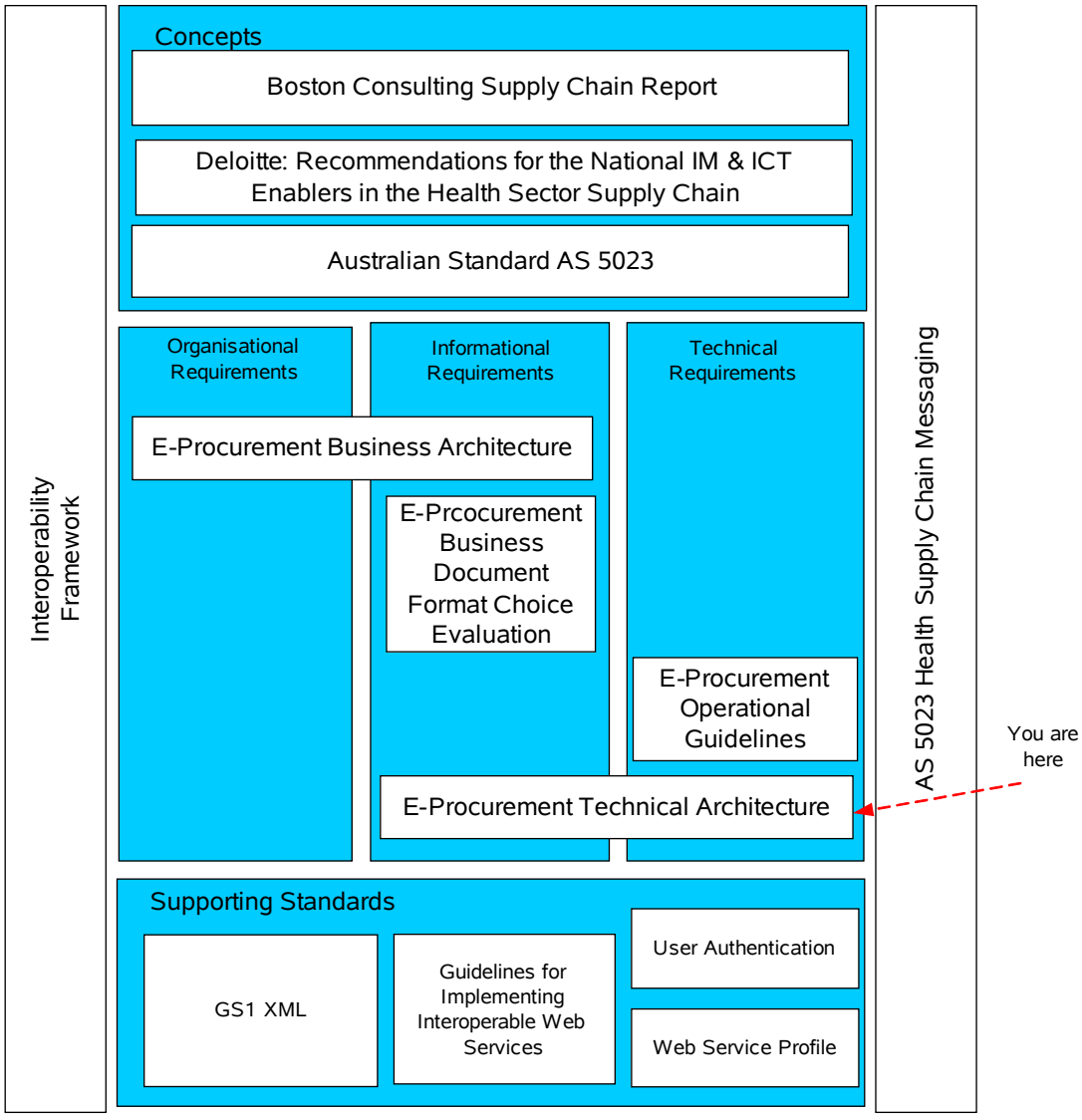

Figure 1: Document Roadmap

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.

1.4 Background

The organisational perspective of this architecture, including the organisational roles and process framework within which E-Procurement messaging is to be conducted, is required reading to understand this technical perspective. It is explained in the E-Procurement Business Architecture [EPBA2007]. 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 [HSCM2004]. 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 a paper entitled *Business Document Format Choices for Health E-Procurement – A Final Evaluation* [BDFCHEP2007]. The full documentation for the GS1 XML documents chosen in this evaluation can be downloaded from [GS1-XML].

The Web Services standards chosen for use in this Implementation Architecture are documented in the NEHTA *Web Services Standards Profile* [WSSP2006]. This profile of standards follows the NEHTA Secure Messaging Architecture [TIAS2006], and is complemented by the *Guidelines for Implementation of Web Services* [GIIWS2007].

1.5 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 2.5 shows some example interactions betweens Buyers, Suppliers and Hubs that illustrate how the Web Services must be used to achieve the desired business communications.

Section 3 introduces the mapping from the informational perspective document definitions, as defined in [HSCM2004], to technical perspective XML document types defined by GS1 XML. It also specifies a number of constraints on the use of the XML in addition to those introduced in [SBDH2007].

Section 4 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. 4.5 XSD and WSDL, shows the full interface descriptions for the Web Services specified.

1.6 Language Used to Describe Imperatives

The keywords MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY in this document are to be interpreted as described in IETF's RFC 2119 [RFC2119].

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`.

The WSDL for each interface is split into two files, following the guidelines in [GIIWS2007]:

- The first file contains Types and PortTypes, and is named `{InvokerRole}(Push|Pull)Interface.wsdl`
- The second file imports the "Interface" file above, and adds the Bindings and Service definitions. It is named `{InvokerRole}(Push|Pull).wsdl`

Please refer to the NEHTA Guidelines for Implementing Web Services [GIIWS2007] for other 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 `BuyerPushInterface.wsdl` file contains a port type called `BuyerPushPortType` which will be able to accept a document of one of the types that Buyers send to their Suppliers using a Push operation (Purchase Order, Purchase Order Change and Purchase Order Cancellation). See Figure 2 for an illustration of this scenario.

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. The parameter type for the generic operation is `NamedDocument`, which contains a string-typed `Name` element to name the logical queue of documents that a Supplier will use in the pull style to request documents of this type using the generic `PullDocument` operation.

NOTE: The generic operation MUST not be used to transmit any of the documents for which specific operations have been designed.

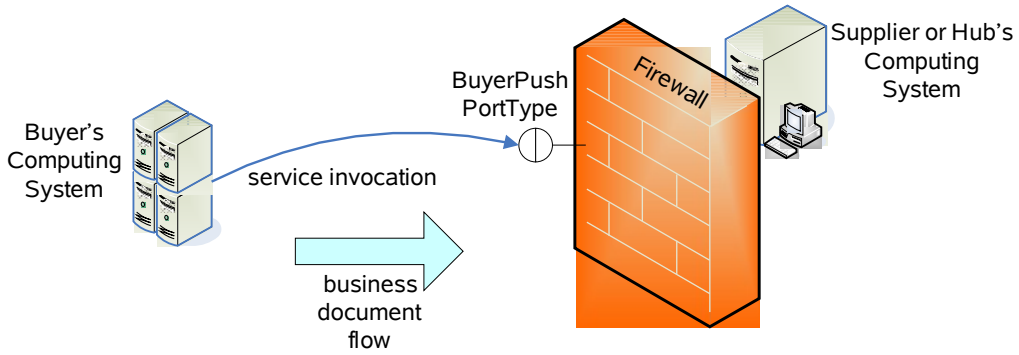

Figure 2: Buyer invokes "push style" interface of Web Service

2.2.2 Pull Model

The other mechanism to get documents from a Buyer to a Supplier via a Hub is to allow Supplier clients of the Hub to poll for new documents rather than supporting a web service interface through their firewall. To facilitate this the `SupplierPullInterface.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 Cancellation).

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 2.

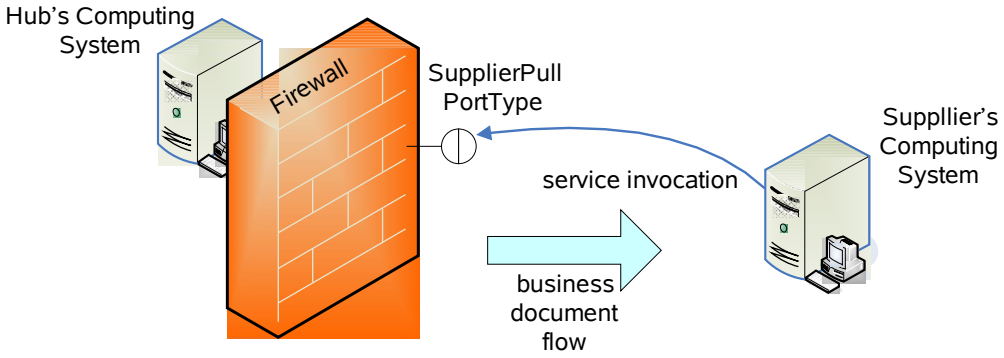

Figure 3: Supplier invokes "pull style" interface of Hub's Web Service

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.

2.2.3 Queue Design for Pull Model

The `QueueDefs` XML Schema defines a data type to return queue information for all queued documents, and data types for each of the Pull operations to return to the Supplier upon invocation.

The queue length information returned from the `GetQueueStatus` operation is defined by the `QueueStatus` type:

```
<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:element name="OrderChangeQueueSize" minOccurs="1"
      maxOccurs="1"
      type="xsd:integer"/>
    <xsd:element name="RCTIQueueSize" 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>
```

Note the unbounded occurrence of the `OtherQueues` element above, which allows for any number of `Name/Size` pairs to be returned, representing the state of queues for documents outside the set selected in the Business Architecture (e.g. Remittance Advice).

The name of the data types defined to return the head of each queue and the remaining queue length follows this template:

`{DocumentType}QueueHead`

The design of these `QueueHead` types allows for the following properties when acting as parameters to the specific-typed operations:

- The name of the document type being retrieved is part of the type name so that the encapsulated document type, which is always wrapped in a `StandardBusinessDocument`, can be known outside the context of the operation.
- The element indicating how many documents remain in the queue, `RemQueueSize`, is mandatory, but the document to be returned from

the queue head is optional, so that the operation can be successfully invoked even when the queue is empty.

Here is an example QueueHead type that returns Orders from the queue of Purchase Orders:

```
<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="sbdh:StandardBusinessDocument"/>
  </xsd:sequence>
</xsd:complexType>
```

There is one other data type defined for use in retrieving any non-NEHTA-standard documents. It is called `OtherQueueHead`, and contains an additional element to represent the name of the queue from which the resulting document is retrieved. The `QueueHead` element is a `NamedDocument` which contains a name to indicate the document type being returned, and an `xsd:anyType` to represent the document. The use of the XML Schema any type allows the maximum flexibility in documents that may be transmitted, although it is recommended that users of this feature wrap their documents in the `StandardBusinessDocument`, as is done for all NEHTA-standard document transmissions.

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

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

2.3 Sending Documents from Hubs or Suppliers to Buyers

2.3.1 Push Model

The only 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 implemented by the Buyer. Similarly, Suppliers using a Hub to send a document to a Buyer will use the "push style" interface supported by the Hub. The `SupplierPushInterface.wsdl` file contains a port type called `SupplierPushPortType` which has operations defined to accept documents of the types that Buyers expect (i.e. Purchase Order Response, Despatch Advice and Invoice).

In addition, the `SupplierPushPortType` interface has a generic `PushDocument` operation to allow buyers to send documents to suppliers that are not nominated in the NEHTA E-Procurement Architecture.

NOTE: The generic operation MUST NOT be used to transmit any of the documents for which specific operations have been designed.

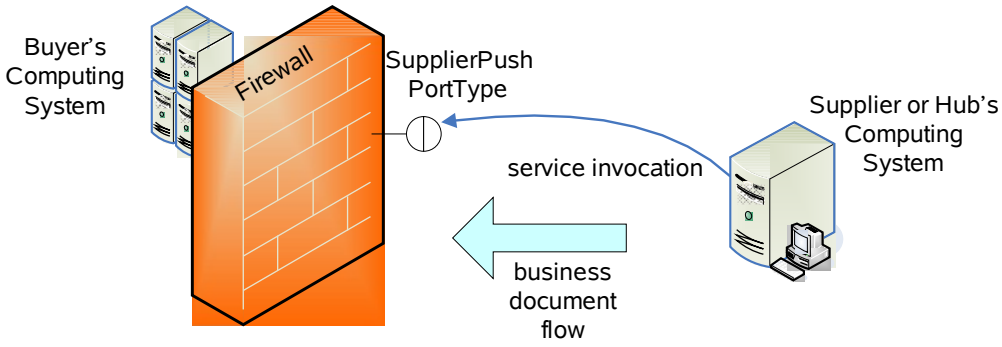

Figure 4: Supplier invokes "push style" interface of Web Service

2.3.2 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 suppliers 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 is 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 2.2.2 for details.

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*.

Hubs that wish to contract for business with Health Departments or publicly funded hospitals are obliged to accept invocations from other hubs using this mechanism. 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 5.

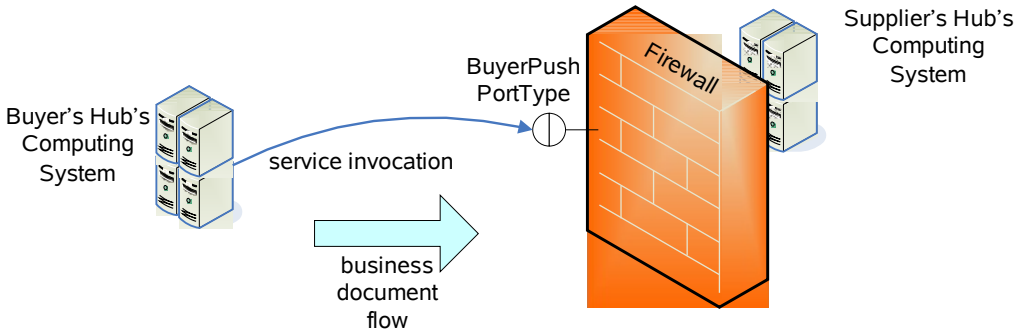

Figure 5: Buyer's Hub interconnects to Supplier's Hub

Conversely, a hub that has received messages via a `SupplierPushPortType` from a supplier, which are destined for a buyer connected to another hub must forward these messages to the buyer's hub using its `SupplierPushPortType`. This is shown in Figure 6.

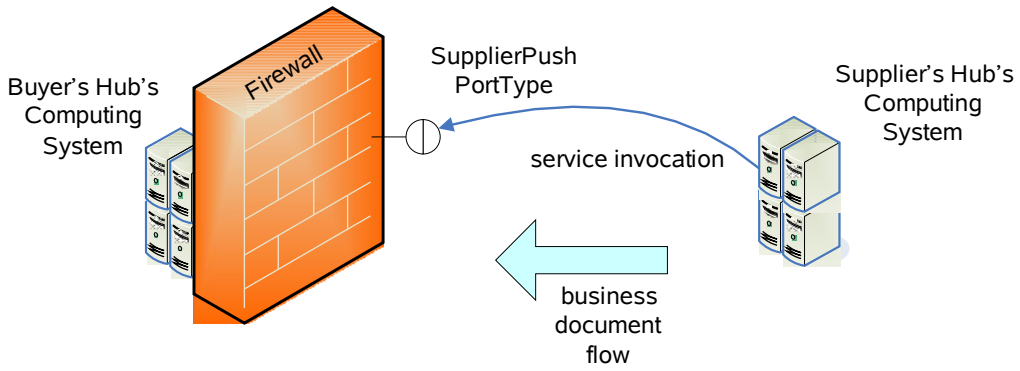

Figure 6: Supplier's Hub interconnects to Buyer's Hub

2.5 Example Interactions

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

2.5.1 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 many 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 its Health 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.

2.5.2 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 SupplierPull 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 2.5.5, 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`.

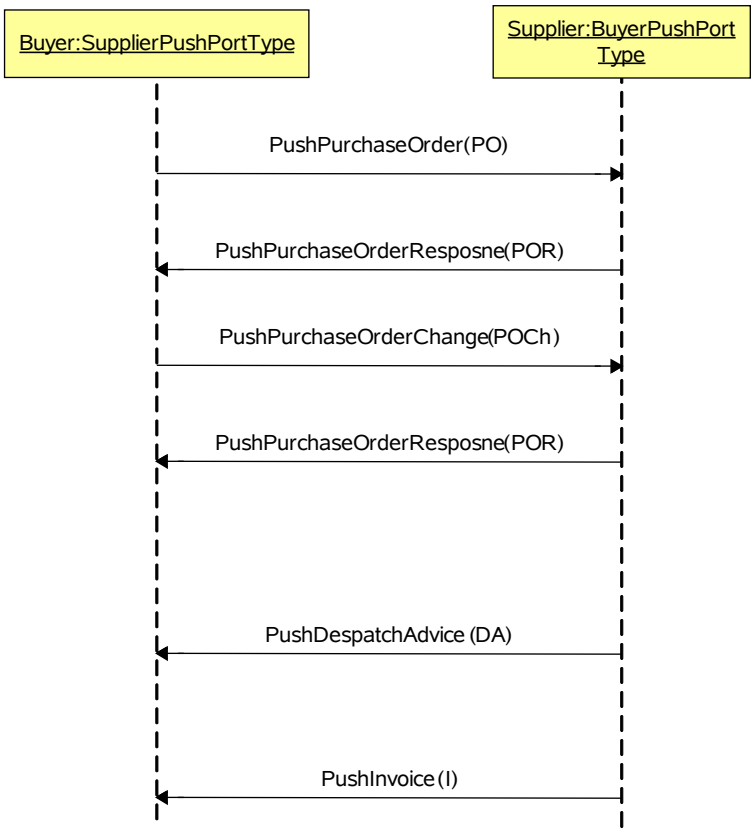

Figure 7: Direct Connect using Push model

The Interaction Diagram in Figure 7 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 a trace of one of the possible instances of the Business Process specified in the E-Procurement Business Architecture [EPBA2007].

2.5.3 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 8 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 9 and Figure 10 for alternative push- and pull-style interactions to deliver these documents to suppliers).

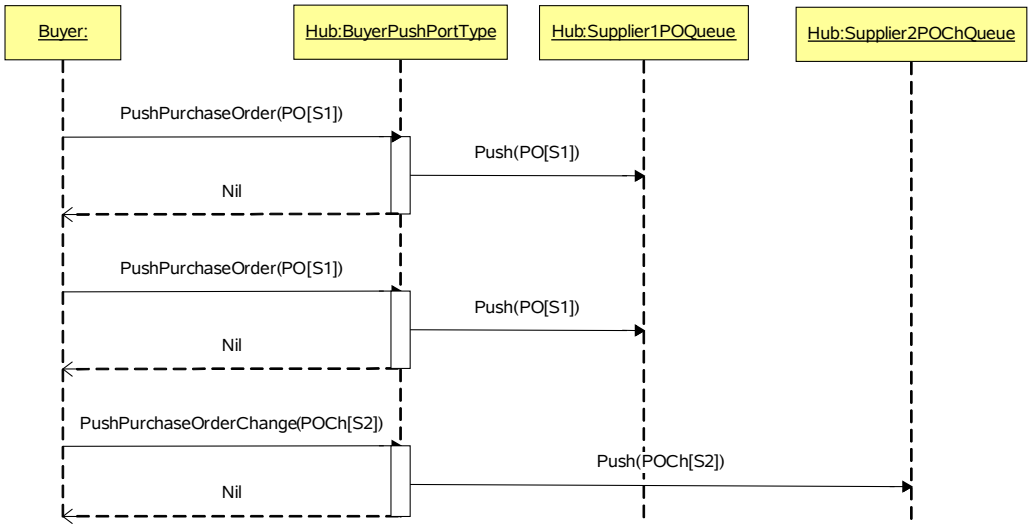

Figure 8: Buyer invokes Push interface at Hub

2.5.4 Buyer Push Interface Supported by Supplier

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 8. (The alternative pull-style delivery is shown in Figure 10.)

Figure 9 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 PO queue for Supplier1, and a single Purchase Order Change placed in the POC queue for Supplier2. These are delivered by invoking the `PushPurchaseOrder` and `PushPurchaseOrderChange` operations, respectively.

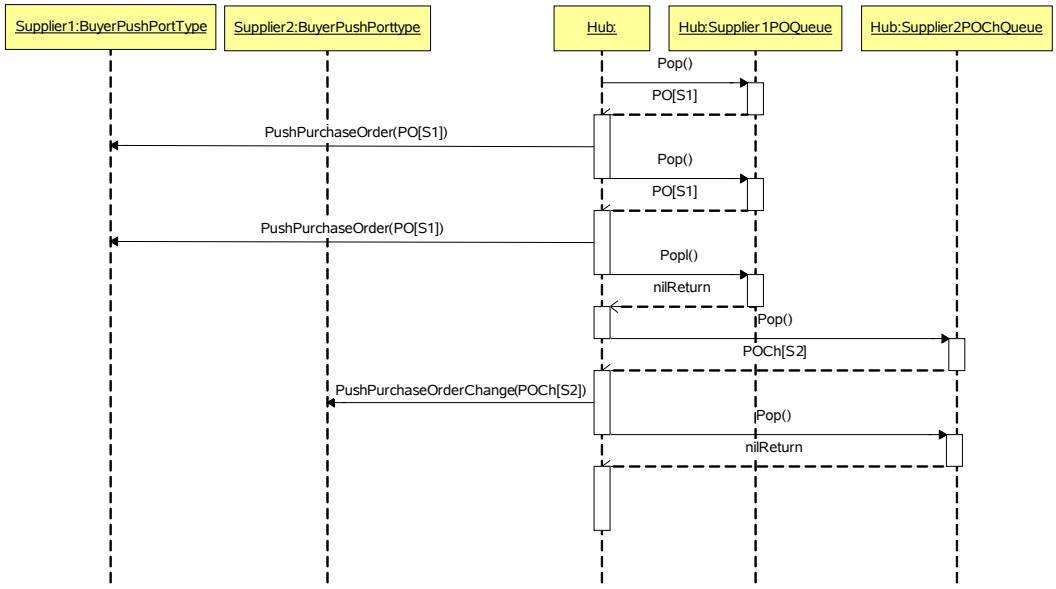

Figure 9: Hub invokes Push interfaces at Suppliers

2.5.5 Supplier Pull Interface Supported by Hub

The example shown in Figure 10 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 8. 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. The `QueueStatus` 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 10, 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`.

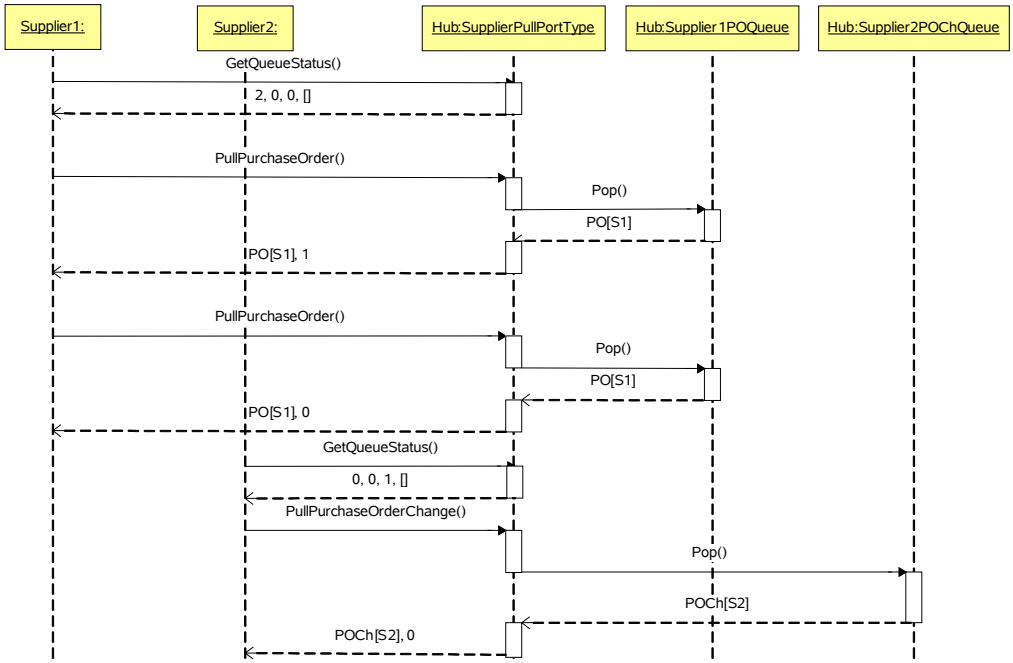

Figure 10: Suppliers invoke Pull interface at Hub

3 Use of GS1 XML

3.1 Introduction

Several XML document formats were evaluated by NEHTA to decide which best suited the purposes of the E-Procurement Architecture. The evaluation is documented in *Business Document Format Choices for Health E-Procurement – A Final Evaluation* [BDFCHEP2007]. The set of XML documents that scored best in the evaluation was GS1 XML.

The GS1 technical message types are always enveloped inside a UN/CEFACT Standard Business Document (SBD). Implementers of this specification must read the UN/CEFACT *Standard Business Document Header Technical Specification* [SBDH2004], and the GS1 document *Standard Business Document Header (SBDH) Version 1.0 Technical Implementation Guide* [SBDH2007] in order to understand the remainder of this section.

This section adds implementation constraints to those specified in the GS1 and UN/CEFACT XML Implementation Guides.

3.2 Mapping between Enterprise Perspective and Technical Perspective

The documents identified from Australian Standard 5023 [HSCM2004] in the informational perspective of the Business Architecture [EPBA2007] for use in the e-procurement process are mapped to the GS1 XML types that will be used to represent them in the technical perspective in . Each GS1 XML document is enveloped in a Standard Business Document (SBD). The GS1 XML Specifications, and their Schema files and implementers packets can be downloaded from [GS1-XML].

AS 5023 Document	GS1 XML Document
Purchase order	SBD enveloping MultiShipmentOrder version 2.4
Purchase order change	SBD enveloping MultiShipmentOrder version 2.4
Purchase order cancellation[1]	SBD enveloping MultiShipmentOrder version 2.4
Purchase order response	SBD enveloping OrderResponse version 2.4
Despatch advice / Advance shipping notice	SBD enveloping DespatchAdvice version 2.4
Invoice	SBD enveloping Invoice version 2.4
Recipient Created Tax Invoice[2]	SBD enveloping Invoice version 2.4

[1] This is not a separate data set in AS5023, which specifies the use of a Purchase order change message for this purpose. Cancellation is identified as a required message in the business process definition, and GS1 XML specifies the use of the MultiShipmentOrder for this purpose.

[2] This is also not a separate data set in AS5023, which specifies the use of a Remittance advice message for this purpose.

Table 1: Mapping from abstract messages to technical messages

3.3 Additional Constraints

The GS1 Implementation Guide for the SBDH [SBDH2007] applies a number of constraints to the use of the StandardBusinessDocument XML Schema for the GS1 context. These MUST be applied for e-procurement documents.

The following additional constraints facilitate interoperability in the narrower context of health e-procurement.

3.3.1 One Document per Invocation

The SBDH XML allows for multiple business documents to be wrapped by and/or attached to a Standard Business Document Header. In the context of health e-procurement there MUST be only one document wrapped in the `documentCommandOperand` element of a SBD, and no documents attached.

3.3.2 Manifest Block

A corollary of the constraint in Section 3.3.1 is that the `Manifest` element, which is used to list attachments to the SBD MUST NOT be used.

3.3.3 Sender and Receiver Blocks

Both `Sender` and `Receiver` elements in the SBDH MUST be populated. The `Identifier` element in each of these MUST use the same GLN information that is contained in the `buyer` and `seller` elements of the wrapped business document.

The `ContactInformation` element of both `Sender` and `Receiver` blocks SHOULD be used. The contact person or people referred to MAY be different from those optionally included in the wrapped business document.

3.3.4 DocumentIdentification Block

The XML element `DocumentIdentification` MUST be populated in all SBDs that are created.

The `Type` element MUST contain the name of the GS1 business document type that is wrapped by the SBD. e.g. "MultiShipmentOrder", or "Invoice".

The `MultipleType` element in the `DocumentIdentification` block MUST be set to "false".

3.3.5 Business Scope Block

The SBD MUST contain exactly one `BusinessScope` element.

This element MUST contain exactly one `Scope` element.

When an SBD envelopes a document identified in Table 1 the `Scope` element's `Type` element MUST contain the string "NEHTA:E-Procurement". In the case where the generic `PushDocument` and `PullDocument` operations are used to transmit documents outside the set identified in Table 1, the `Type` element MAY contain some other process name of shared significance to the trading partners.

When an SBD envelopes a document identified in Table 1 the `Scope` element's `InstanceIdentifier` element MUST contain the version number of the Message Implementation Guideline for the wrapped document that was used as the standard to generate the XML mapping for the message.

In the case where the generic `PushDocument` and `PullDocument` operations are used to transmit documents outside the set identified in Table 1, the

`InstanceIdentifier` element MAY contain an identifier of the wrapped document.

The `Scope` element's `Identifier` element MAY contain other information by agreement between the trading partners, or it MAY be absent.

The nested `BusinessService` and `CorrelationInformation` elements MUST NOT be present.

3.3.6 The Message Element

When an SBD envelopes a document identified in Table 1 the `message` element's `uniqueCreatorIdentification` element MUST contain the `uniqueCreatorIdentification` of the `orderIdentification`, `despatchAdviceIdentification`, or `invoiceIndetification` element of the wrapped document.

4 Service Definitions

This section documents the major XML argument types and Port Types defined in the WSDL files: `BuyerPush.wsdl` (which imports `BuyerPushInterface.wsdl`), `SupplierPush.wsdl` (which imports `SupplierPushInterface.wsdl`) and `SupplierPull.wsdl` (which imports `SupplierPullInterface.wsdl`). There are also two XSD files which define some types needed by the WSDL: `Nil.xsd` and `QueueTypes.xsd`. These files are listed in full in 4.5XSD and WSDL.

4.1 Namespace Prefixes

The XSD and WSDL defined for e-procurement consistently use the following namespace prefixes:

`tns`	the current targetNamespace
`nilns`	`"urn:xml-gov-au:nehta:types:common:Nil:1.0"`
`wsaw`	`"http://www.w3.org/2006/05/addressing/wsdl"`
`sbdh`	`"http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"`
`queuens`	`"urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0"`

4.2 Common Type Definitions

Type Definitions

Other than the GS1 document types and the UN/CEFACT Standard Business Document Header, there is only one XML type definition that is used by all port types – the `Nil` type, found in `types/common/Nil.xsd`, and in Appendix 4.5.

```
<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.

4.3 BuyerPushPortType

Operation: PushPurchaseOrder

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0`

Preconditions

None.

Input

```
xsd:element name="PushPurchaseOrder"
type="sbdh:StandardBusinessDocument"
```

The documentCommandHeader element of the Standard Business Document MUST have its type attribute set to "ADD".

```
sh:StandardBusinessDocument
  eanucc:message
    eanucc:transaction
      command
        eanucc:documentCommand
          documentCommandHeader type="ADD"
```

The gln element nested in the orderPartyInformation element's buyer element MUST be that of the organisational entity making the order. The combination of this GLN and the purchase order identifier (as given in the Multi Shipment Order element orderIdentification/uniqueCreatorIdentification) act as the unique key to identifying the Order in all subsequent documents relating to it in the procurement process being enacted.**Output**

```
xsd:element name="PushPurchaseOrderResponse"
type="nilns:Nil"
```

Behaviour

PushPurchaseOrder transmits a Standard Business Document which contains a Multi Shipment Order as a child of its documentCommandOperand element. The document flows from a buyer to a supplier or hub, returning a Nil upon successful completion.

Operation: PushPurchaseOrderCancel

Target Namespace

urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0

Preconditions

An order must have already been sent.

Input

```
xsd:element name="PushPurchaseOrderCancel"
type="sbdh:StandardBusinessDocument"
```

The documentCommandHeader element of the Standard Business Document which has the Multi Shipment Order as its documentCommandOperand MUST have its type attribute set to "DELETE".

```
sh:StandardBusinessDocument
  eanucc:message
    eanucc:transaction
      command
        eanucc:documentCommand
          documentCommandHeader type="DELETE"
```

The documentOperand of the SBD must contain a MultiShipmentOrder instance.

The gln element nested in the Multi Shipment Order's orderPartyInformation/buyer element must be the same as that in the order being cancelled. This GLN and the original purchase order's identifier (as given in the Multi Shipment Order's element orderIdentification/uniqueCreatorIdentification) together act as the unique key to identifying the original order to be cancelled.

Output

```
xsd:element name="PushPurchaseOrderCancelResponse"
type="nilns:Nil"
```

Behaviour

PushPurchaseOrderCancel transmits a GS1 Multi Shipment Order document wrapped inside a Standard Business Document from a buyer to a supplier or hub, returning a Nil upon successful completion. The recipient supplier will use the document as detailed above to identify an existing order that must be cancelled.

Operation: PushPurchaseOrderChange

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0`

Preconditions

A purchase order must already have been sent to a supplier, and a purchase order response relating to the order must have been received from that supplier.

Input

```
name="PushPurchaseOrderChange"
type="sbdh:StandardBusinessDocument"
```

The `documentCommandHeader` element of the Standard Business Document which has the Multi Shipment Order as its `documentCommandOperand` must have its type attribute set to `"CHANGE_BY_REFRESH"`.

```
sh:StandardBusinessDocument
  eanucc:message
    eanucc:transaction
      command
        eanucc:documentCommand
          documentCommandHeader
          type="CHANGE_BY_REFRESH"
```

The documentOperand of the SBD must contain a MultiShipmentOrder instance.

The `gln` element nested in the MultiShipmentOrder's `orderPartyInformation/buyer` element must be the same as that in the order being changed. This GLN and the original purchase order's identifier (as given in the Multishipment Order's element `orderIdentification/uniqueCreatorIdentification)` act as the unique key to identifying the original order to be changed.

The line items in the Multi Shipment Order wrapped by the SBD parameter MUST each include the optional element `lineItemActionCode` to indicate whether the line is unchanged (set to `"NOT_AMENDED"`), changed (set to `"CHANGED"`), deleted (set to `"DELETED"`), or added (set to `"ADDITION"`). All line items from the original Order MUST be in the changed Order, with the same line numbers, and any new lines MUST be added at the end.

Output

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

Behaviour

PushPurchaseOrderChange transmits a Multi Shipment Order document from a buyer to a supplier or hub, returning a Nil upon successful

completion. The recipient supplier will use the document as detailed above to identify an existing order that must be changed.

Operation: PushRCTI

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0`

Preconditions

None.

Input

```
xsd:element name="PushRCTI"
type="sbdh:StandardBusinessDocument"
```

The `invoiceType` element of the invoice wrapped in the SBD must contain the invoice code "SELF_BILLED_INVOICE".

The `documentCommandHeader` element of the Standard Business Document MUST have its type attribute set to `"ADD"`.

```
sh:StandardBusinessDocument
  eanucc:message
    eanucc:transaction
      command
        eanucc:documentCommand
          documentCommandHeader type="ADD"
```

The `gln` element nested in the `orderPartyInformation` element's `buyer` element MUST be that of the organisational entity making the order.

Output

```
xsd:element name="PushRCTIResponse" type="nilns:Nil"
```

Behaviour

`PushRCTI` transmits a Standard Business Document which contains an Invoice as a child of its `documentCommandOperand` element. The document flows from a buyer to a supplier or hub, returning a Nil upon successful completion.

Operation: PushDocument

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0`

Preconditions

None.

Input

```
name="PushDocument" type="queuens:NamedDocument"
```

Output

```
name="PushDocumentResponse" type="nilns:Nil"
```

Behaviour

`PushDocument` transmits a document from a buyer to a supplier or hub as an XML any type in the `Document` element of the `NamedDocument` for a queue named in the `QueueName` element of the NamedDocument, returning a Nil upon successful completion.

It is strongly recommended that a Standard Business Document (SBD) be used to wrap the business document being transmitted in the same way as

this wrapper is used in the well typed operations. When a SBD is used as the parameter type, the addressee of the document will be the Supplier nominated by its GLN in the `Receiver` element of the Standard Business Document Header. If any other type is used, then the convention for extracting the addressee details will need to be agreed between the trading partners and their hubs.

The `NamedDocument`'s `QueueName` element will be the same as that used as the name of the queue returned by a `GetQueueStatus` operation on the SupplierPull port type.

4.4 SupplierPushPortType

Operation: PushPurchaseOrderResponse

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0`

Preconditions

A Purchase Order must have been received from a buyer.

Input

```
name="PushPurchaseOrderResponse"
type="sbdh:StandardBusinessDocument"
```

The `gln` element nested in the `OrderResponse`'s `buyer` element must be the same as that in the order being changed. This GLN and the original purchase order identifier (as given in the attribute `responseToOriginalDocument.referenceIdentification)` act as the unique key to identifying the original order.

A "Positive Purchase Order Response", as shown in the process model in the Business Architecture [EPBA2007] is indicated by using the attribute value responseStatusType="ACCEPTED" in the OrderResponse element.

A "Negative Purchase Order Response", as shown in the process model can either be

- a rejection of the entire order, which is indicated by using the attribute value responseStatusType="REJECTED" in the OrderResponse element,
- or a suggested amendment to the order, which is indicated by using the attribute value responseStatusType="MODIFIED". In this case the Order Response will detail the proposed change, and the supplier will await a Purchase Order Change or Purchase Order Cancellation.

Output

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

Behaviour

PushPurchaseOrderResponse transmits a Standard Business Document which wraps an Order Response document from a supplier to a buyer or hub, returning a Nil upon successful completion.

Operation: PushDespatchAdvice

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0`

Preconditions

None.

Input

```
name="PushDespatchAdvice"
type="sbdh:StandardBusinessDocument"
```

Output

```
name="PushDespatchAdviceResponse" type="nilns:Nil"
```

Behaviour

PushDespatchAdvice transmits a Standard Business Document containing a wrapped DespatchAdvice document from a buyer to a supplier or hub, returning a Nil upon successful completion.

Operation: PushInvoice

Target Namespace

```
urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0
```

Preconditions

None.

Input

```
name="PushInvoice"
type="sbdh:StandardBusinessDocument"
```

Output

```
name="PushInvoiceResponse" type="nilns:Nil"
```

Behaviour

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

Operation: PushDocument

Target Namespace

```
urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0
```

Preconditions

None.

Input

```
name="PushDocument" type="queuens:NamedDocument"
```

Output

```
name="PushDocumentResponse" type="nilns:Nil"
```

Behaviour

PushDocument transmits a document as an xsd:any for a named queue from a supplier to a buyer or hub, returning a Nil upon successful completion.

It is strongly recommended that a Standard Business Document (SBD) be used to wrap the business document being transmitted in the same way as this wrapper is used in the well typed operations. When a SBD is used as the parameter type, the addressee of the document will be the Buyer nominated by its GLN in the `Receiver` element of the Standard Business Document Header. If any other type is used, then the convention for extracting the addressee details will need to be agreed between the trading partners and their hubs.

4.5 SupplierPullPortType

The pull paradigm is designed to suit Suppliers that cannot, or do not wish to implement push-style Web Services interfaces, and make them available for invocation through their firewalls. The SupplierPull port type will be implemented and made accessible by Hubs for Suppliers to invoke. By necessity a polling approach is required, as there is no standard mechanism to alert Suppliers to the availability of new e-procurement documents for them to collect. As such, the queues of messages that hubs may have available for collection must be exposed through an interface. The `GetQueueStatus` operation returns an instance of the XML data type `QueueStatus`, which informs the Supplier how many documents, of which types, are available for pulling. `QueueStatus` also allows the Hub to reveal the status of from zero to an unlimited number of additional named queues, which can be used to transmit documents outside the scope of this architecture.

The ordering of document transmissions in the Procurement Public Process defined in the Business Architecture [EPBA2007] has a dependency between Purchase Order and the subsequent documents Purchase Order Change and Purchase Order Cancellation. It is therefore important that the Supplier pull the messages from the queues in that order, so that all Change and Cancellation documents can be processed after the Orders to which they refer.

DataType: QueueStatus

Target Namespace

`urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0`

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: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.

Example Instance

The example instance below indicates that the following documents are waiting to be retrieved using invocations on the SupplierPull interface:

- 3 Orders
- 1 Order Cancellation
- 0 Order Changes
- 0 Recipient Created Tax Invoices
- 2 Documents on the "Quotation" named queue
- 1 Document on the "RemittanceAdvice" named queue

```
<?xml version="1.0" encoding="UTF-8"?>
<qd:QueueStatus xmlns:qd="urn:xml-gov-au:nehta:types:supply-
chain:QueueDefs:1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="QueueStatus.xsd">
    <qd:OrderQueueSize>3</qd:OrderQueueSize>
    <qd:OrderCancelQueueSize>1</qd:OrderCancelQueueSize>
    <qd:OrderChangeQueueSize>0</qd:OrderChangeQueueSize>
    <qd:RCTIQueueSize>0</qd:RCTIQueueSize>
    <qd:OtherQueues>
        <qd:Name>Quotation</qd:Name>
        <qd:Size>2</qd:Size>
    </qd:OtherQueues>
    <qd:OtherQueues>
        <qd:Name>RemittanceAdvice</qd:Name>
        <qd:Size>1</qd:Size>
    </qd:OtherQueues>
</qd:QueueStatus>
```

DataType: OrderQueueHead

Target Namespace

`urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0`

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="sbdh:StandardBusinessDocument"/>
  </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. `QueueHead` is optional so that correct invocations can be made on Pull operations even when there are zero documents in the queue.

DataType: OrderCancelQueueHead

Target Namespace

`urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0`

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="sbdh:StandardBusinessDocument"/>
  </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. `QueueHead` is optional so that correct invocations can be made on Pull operations even when there are zero documents in the queue.

DataType: OrderChangeQueueHead

Target Namespace

`urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0`

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="sbdh:StandardBusinessDocument"/>
  </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. QueueHead is optional so that correct invocations can be made on Pull operations even when there are zero documents in the queue.

DataType: RCTIQueueHead

Target Namespace

urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0

Definition

```
<xsd:complexType name="RCTIQueueHead">
  <xsd:sequence>
    <xsd:element name="RemQueueSize" minOccurs="1"
           maxOccurs="1" type="xsd:integer"/>
    <xsd:element name="QueueHead" minOccurs="0"
           maxOccurs="1"
           type="sbdh:StandardBusinessDocument"/>
  </xsd:sequence>
</xsd:complexType>
```

Purpose

The RCTIQueueHead type is the return type for the PullRCTI operation. It contains two elements: RemQueueSize, which indicates how many documents remain on the Recipient Created Tax Invoice queue at the hub, and QueueHead which is the invoice document that has just been popped from the head of the queue. QueueHead is optional so that correct invocations can be made on Pull operations even when there are zero documents in the queue.

DataType: OtherQueueHead

Target Namespace

urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0

Definition

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

Purpose

The OtherQueueHead type is the return type for the generic PullDocument operation. It contains a Name element which names the

queue, a `RemQueueSize` element which is an integer indicating how many documents remain in the named queue, and a `QueueHead` element which is of type `StandardBusinessDocument` for the document that has just been popped off the named queue. `QueueHead` is optional so that correct invocations can be made on Pull operations even when there are zero documents in the queue.

Operation: GetQueueStatus

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0`

Preconditions

None.

Input

```
name="GetQueueStatus" type="nilns:Nil"
```

Output

```
name="GetQueueStatusResponse"
type="queuens: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

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0`

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="nilns:Nil"
```

Output

```
name="PullPurchaseOrderResponse"
type="queuens: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

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0`

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="nilns:Nil"
```

Output

```
name="PullPurchaseOrderCancelResponse"
type="queuens: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

```
urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0
```

Preconditions

A Purchase Order Response must have been sent by the supplier to respond to an Order from a buyer before the buyer may send a Purchase Order Change relating to the Order.

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="nilns:Nil"
```

Output

```
name="PullPurchaseOrderChangeResponse"
type="queuens: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: PullRCTI

Target Namespace

```
urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0
```

Preconditions

In order for a Recipient Created Tax Invoice 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 invoice documents queued, but the invocation will return no document.

Input

`name="PullRCTI" type="nilns:Nil"`

Output

`name="PullRCTIResponse" type="queuens:RCTIQueueHead"`

Behaviour

`PullRCTI` has a `Nil` typed input. It returns an `RCTIQueueHead` element, which contains the invoice at the head of the queue for the invoking Supplier, if the queue is non-empty. The other component of the `RCTIQueueHead` is the remaining queue size, indicating how many more Recipient Created Tax Invoices are still to be pulled by repeated invocations of this operation.

Operation: PullDocument

Target Namespace

`urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0`

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="queuens:QueueName"`

Output

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

Fault

```
name="UnknownQueueNameFault"
type="queuens:UnknownQueueName"
```

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 `any` type 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. The invocation will fail with an `UnknownQueueNameFault` if the `QueueName` passed in the input parameter is unknown to the service.

Appendix A:Glossary

Activity Diagram	A UML model and diagramming standard for representing processes and data flows.
AS2	Applicability Statement 2 is a specification about how to transport data securely and reliably over the Internet. It is specified by IETF in RFC 4130. It supports encryption and digital signing, and non-repudiation. It uses HTTP over TCP/IP or SSL.
CSV	Comma Separated Value. A text file format where table column entries are separated by commas, and rows by newlines. This format is often exported from spreadsheets and databases.
EDI	Electronic Data Interchange. A late 1970s initiative to interchange documents between business partners in electronic formats.
GS1	Global Standards 1. A not-for-profit global standards body for bar codes, product data synchronisation and RFID technologies.
HTTP	Hypertext Transfer Protocol. A standard protocol for transferring documents - used in the World Wide Web, and Web Services.
IETF	Internet Engineering Task Force. The body that standardises much of the protocol infrastructure for the internet.
MIME	Multipurpose Internet Mail Extensions
RCTI	Recipient Created Tax Invoice. An invoice created by the buyer based on agreed prices and charges and sent to the supplier.
SOAP	Simple Object Access Protocol. A protocol for exchanging XML-based messages over computer network. A fundamental part of Web Services.
SSL	Secure Sockets Layer. A cryptographic protocol for private communications over the internet.
TCP/IP	The fundamental transport protocols of the internet, over which all other application layer protocols (such as SSL and HTTP) are transmitted.
UBL	Universal Business Language. A set of

	XML documents designed at OASIS for business interchange. Includes all of the e-procurement documents.
UMM	UN/CEFACT Modelling Methodology.
UML	Unified Modelling Language.
VAN	Value Added Network. A store and forward service for transmitting EDI messages between business partners. Historically this was done with modems or leased lines, but now VANs operate over the internet.
WSDL	Web Services Description Language.
XML	eXtensible Markup Language.
XML Schema	A language used to prescribe formatting rules for XML documents.
XSD	XML Schema Description. The file type for XML Schema documents.
UN/EDIFACT	United Nations/Electronic Data Interchange For Administration, Commerce, and Transport. A set of standards established by the UN Centre for Trade Facilitation and Electronic Business.

Appendix B:References

[BDFCHEP2007] NEHTA, *Business Document Format Choices for Health E-Procurement - A Final Evaluation*, 2007.

[EDIFACT-UN] United Nations Economic Commission for Europe, 13 December 2006, *United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport*, accessed 28 June 2007, <http://www.unece.org/trade/untdid/welcome.htm>

[EPBA2007] NEHTA, *E-Procurement Business Architecture*, Version 1.0, June 2007.

[EPOG2007] NEHTA, *E-Procurement Operational Guidelines*, Version 1.0, June 2007.

[GIIWS2007] NEHTA, *Guidelines for Implementing Interoperable Web Services*, version 1.0, 2007.

[GS1-XML] GS1, *GS1 XML - Technical*, accessed 28 June 2007, <http://gs1.org/productssolutions/ecom/xml/technical/>

[HSCM2004] Standards Australia, Australian Standard 5023.[1-4], *Health Supply Chain Messaging*, 19 March 2004.

[NIF2006] NEHTA, *Interoperability Framework*, 2006.

[TAIS2006] NEHTA, *Technical Architecture for Implementing Services* v1.0, December 2006.

[SBDH2004] UN/CEFACT, *UN/CEFACT Standard Business Document Header*, Technical Specification, Version 1.3, 4 June 2004.

[SBDH2007] GS1, *Standard Business Document Header (SBDH)*, Version 1.0, Technical Implementation Guide, Draft 0.3, May 2007.

[WSSP2006] NEHTA, *Web Services Standards Profile*, version 2.0, 2006.

Appendix C:XSD and WSDL

C.1 Nil.xsd

```
<?xml version="1.0"?>
<xsd:schema elementFormDefault="qualified"
  targetNamespace="urn:xml-gov-au:nehta:types:common:Nil:1.0"
  xmlns:nilns="urn:xml-gov-au:nehta:types:common:Nil:1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

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

</xsd:schema>
```

C.2 QueueDefs.xsd

```
<?xml version="1.0"?>
<xsd:schema elementFormDefault="qualified"
  targetNamespace="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0"
  xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sbdh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
>

  <xsd:import
namespace="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
    schemaLocation="../../../GS1_Consolidated_XML/sbdh/StandardBusinessDocumentHeader.xsd"/>

  <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:element name="OrderChangeQueueSize" minOccurs="1" maxOccurs="1"
        type="xsd:integer"/>
      <xsd:element name="RCTIQueueSize" 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>

  <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="sbdh:StandardBusinessDocument"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="OrderCancelQueueHead">
    <xsd:sequence>
      <xsd:element name="RemQueueSize" minOccurs="1" type="xsd:integer"/>
      <xsd:element name="QueueHead" minOccurs="0"
        type="sbdh:StandardBusinessDocument"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="OrderChangeQueueHead">
    <xsd:sequence>
      <xsd:element name="RemQueueSize" minOccurs="1" type="xsd:integer"/>
      <xsd:element name="QueueHead" minOccurs="0"
        type="sbdh:StandardBusinessDocument"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="RCTIQueueHead">
    <xsd:sequence>
      <xsd:element name="RemQueueSize" minOccurs="1" type="xsd:integer"/>
      <xsd:element name="QueueHead" minOccurs="0"
        type="sbdh:StandardBusinessDocument"/>
    </xsd:sequence>
  </xsd:complexType>

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

```
  </xsd:complexType>

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

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

  <xsd:complexType name="UnknownQueueName">
    <xsd:sequence>
      <xsd:element name="UnknownQueueName" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

C.3 BuyerPushInterface.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="BuyerPush"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0"

  xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0"

  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"

  xmlns:sbdh="http://www.unece.org/cefact/namespaces/StandardBusinessDocum
entHeader"

>


  <!-- ====== Type Declarations ====== -->


  <wsdl:types>

    <xsd:schema elementFormDefault="qualified"

      targetNamespace="urn:xml-gov-au:nehta:service:supply-
chain:BuyerPush:1.0"

      xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0"

      xmlns:nilns="urn:xml-gov-au:nehta:types:common:Nil:1.0"

      xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:
1.0"

      xmlns:xsd="http://www.w3.org/2001/XMLSchema">


      <!-- ====== Imports ====== -->

      <xsd:import namespace="urn:xml-gov-au:nehta:types:common:Nil:1.0"

        schemaLocation="../../types/common/Nil.xsd"/>

      <xsd:import namespace="urn:xml-gov-au:nehta:types:supply-
chain:QueueDefs:1.0"

        schemaLocation="../../types/supply-chain/QueueDefs.xsd"/>

      <xsd:import
namespace="http://www.unece.org/cefact/namespaces/StandardBusinessDocument
Header"

        schemaLocation="../../../GS1_Consolidated_XML/sbdh/StandardBusines
sDocumentHeader.xsd"/>


      <!-- ====== Element Declarations ====== -->

      <xsd:element name="PushPurchaseOrder"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushPurchaseOrderCancel"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushPurchaseOrderChange"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushRCTI" type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushDocument" type="queuens:NamedDocument"/>

      <xsd:element name="PushPurchaseOrderResponse" type="nilns:Nil"/>
```

```
      <xsd:element name="PushPurchaseOrderCancelResponse"
type="nilns:Nil"/>
      <xsd:element name="PushPurchaseOrderChangeResponse"
type="nilns:Nil"/>
      <xsd:element name="PushRCTIResponse" type="nilns:Nil"/>
      <xsd:element name="PushDocumentResponse" type="nilns:Nil"/>
    </xsd:schema>
  </wsdl:types>

  <!-- ====== Message Types ====== -->

  <wsdl:message name="PushPurchaseOrderInMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrder"/>
  </wsdl:message>

  <wsdl:message name="PushPurchaseOrderOutMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrderResponse"/>
  </wsdl:message>

  <wsdl:message name="PushPurchaseOrderCancelInMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrderCancel"/>
  </wsdl:message>

  <wsdl:message name="PushPurchaseOrderCancelOutMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrderCancelResponse"/>
  </wsdl:message>

  <wsdl:message name="PushPurchaseOrderChangeInMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrderChange"/>
  </wsdl:message>

  <wsdl:message name="PushPurchaseOrderChangeOutMsg">
    <wsdl:part name="body" element="tns:PushPurchaseOrderChangeResponse"/>
  </wsdl:message>

  <wsdl:message name="PushRCTIInMsg">
    <wsdl:part name="body" element="tns:PushRCTI"/>
  </wsdl:message>

  <wsdl:message name="PushRCTIOutMsg">
    <wsdl:part name="body" element="tns:PushRCTIResponse"/>
  </wsdl:message>

  <wsdl:message name="PushDocumentInMsg">
    <wsdl:part name="body" element="tns:PushDocument"/>
  </wsdl:message>
```

```
  <wsdl:message name="PushDocumentOutMsg">
    <wsdl:part name="body" element="tns:PushDocumentResponse"/>
  </wsdl:message>

  <!-- ====== Port Type Declarations ====== -->

  <wsdl:portType name="BuyerPushPortType">

    <wsdl:operation name="PushPurchaseOrder">
      <wsdl:input message="tns:PushPurchaseOrderInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrder"/>
      <wsdl:output message="tns:PushPurchaseOrderOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PushPurchaseOrderCancel">
      <wsdl:input message="tns:PushPurchaseOrderCancelInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderCancel"/>
      <wsdl:output message="tns:PushPurchaseOrderCancelOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderCancelResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PushPurchaseOrderChange">
      <wsdl:input message="tns:PushPurchaseOrderChangeInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderChange"/>
      <wsdl:output message="tns:PushPurchaseOrderChangeOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderChangeResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PushRCTI">
      <wsdl:input message="tns:PushRCTIInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushRCTI"/>
      <wsdl:output message="tns:PushRCTIOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushRCTIResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PushDocument">
      <wsdl:input message="tns:PushDocumentInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushDocument"/>
```

```
      <wsdl:output message="tns:PushDocumentOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushDocumentResponse"/>
    </wsdl:operation>


  </wsdl:portType>


</wsdl:definitions>
```

C.4 BuyerPush.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="BuyerPush"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:1.0"

  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"

>


  <!-- ====== WSDL Import ====== -->


  <wsdl:import namespace="urn:xml-gov-au:nehta:service:supply-
chain:BuyerPush:1.0"

    location="BuyerPushInterface.wsdl"/>


  <!-- ====== Binding Definitions ====== -->


  <wsdl:binding name="BuyerPushSOAPBinding" type="tns:BuyerPushPortType">

    <soap:binding style="document"

      transport="http://schemas.xmlsoap.org/soap/http"/>


    <wsdl:operation name="PushPurchaseOrder">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrder"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>

    </wsdl:operation>


    <wsdl:operation name="PushPurchaseOrderCancel">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderCancel"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>
```

```
    </wsdl:operation>

    <wsdl:operation name="PushPurchaseOrderChange">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushPurchaseOrderChange"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="PushRCTI">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushRCTI"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="PushDocument">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-chain:BuyerPush:
1.0/PushDocument"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>

  </wsdl:binding>

  <!-- ====== Service Definitions ====== -->

  <wsdl:service name="BuyerPush">
    <wsdl:documentation>Web Service for Buyers to invoke to transmit
      e-procurement documents using Push model</wsdl:documentation>
    <wsdl:port binding="tns:BuyerPushSOAPBinding" name="BuyerPushPort">
      <soap:address location="JURISDICTION_BINDING_URL"/>
```

```
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

C.5 SupplierPushInterface.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="SupplierPushInterface"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:1.0"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0"

  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"

  xmlns:sbdh="http://www.unece.org/cefact/namespaces/StandardBusinessDocum
entHeader"

>



  <wsdl:types>

    <xsd:schema elementFormDefault="qualified"

      targetNamespace="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0"

      xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:
1.0"

      xmlns:nilns="urn:xml-gov-au:nehta:types:common:Nil:1.0"

      xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:
1.0"

      xmlns:xsd="http://www.w3.org/2001/XMLSchema">



      <!-- ====== XSD Imports ====== -->

      <xsd:import namespace="urn:xml-gov-au:nehta:types:common:Nil:1.0"

             schemaLocation="../../types/common/Nil.xsd"/>

      <xsd:import namespace="urn:xml-gov-au:nehta:types:supply-
chain:QueueDefs:1.0"

             schemaLocation="../../types/supply-chain/QueueDefs.xsd"/>

      <xsd:import
namespace="http://www.unece.org/cefact/namespaces/StandardBusinessDocument
Header"

        schemaLocation="../../../GS1_Consolidated_XML/sbdh/StandardBusines
sDocumentHeader.xsd"/>



      <!-- ====== Element Definitions ====== -->

      <xsd:element name="PushPurchaseOrderResponse"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushDespatchAdvice"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushInvoice"
type="sbdh:StandardBusinessDocument"/>

      <xsd:element name="PushPurchaseOrderResponseResponse"
type="nilns:Nil"/>

      <xsd:element name="PushDespatchAdviceResponse" type="nilns:Nil"/>

      <xsd:element name="PushInvoiceResponse" type="nilns:Nil"/>

      <xsd:element name="PushDocument" type="queuens:NamedDocument"/>
```

```
        <xsd:element name="PushDocumentResponse" type="nilns:Nil"/>
      </xsd:schema>
    </wsdl:types>

    <!-- ====== Message Definitions ====== -->

    <wsdl:message name="PushPurchaseOrderResponseInMsg">
      <wsdl:part name="body" element="tns:PushPurchaseOrderResponse"/>
    </wsdl:message>

    <wsdl:message name="PushPurchaseOrderResponseOutMsg">
      <wsdl:part name="body"
element="tns:PushPurchaseOrderResponseResponse"/>
    </wsdl:message>

    <wsdl:message name="PushDespatchAdviceInMsg">
      <wsdl:part name="body" element="tns:PushDespatchAdvice"/>
    </wsdl:message>

    <wsdl:message name="PushDespatchAdviceOutMsg">
      <wsdl:part name="body" element="tns:PushDespatchAdviceResponse"/>
    </wsdl:message>

    <wsdl:message name="PushInvoiceInMsg">
      <wsdl:part name="body" element="tns:PushInvoice"/>
    </wsdl:message>

    <wsdl:message name="PushInvoiceOutMsg">
      <wsdl:part name="body" element="tns:PushInvoiceResponse"/>
    </wsdl:message>

    <wsdl:message name="PushDocumentInMsg">
      <wsdl:part name="body" element="tns:PushDocument"/>
    </wsdl:message>

    <wsdl:message name="PushDocumentOutMsg">
      <wsdl:part name="body" element="tns:PushDocumentResponse"/>
    </wsdl:message>

    <!-- ====== Port Type Definitions ====== -->

    <wsdl:portType name="SupplierPushPortType">

      <wsdl:operation name="PushPurchaseOrderResponse">
        <wsdl:input message="tns:PushPurchaseOrderResponseInMsg"
```

```
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushPurchaseOrderResponse"/>

      <wsdl:output message="tns:PushPurchaseOrderResponseOutMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushPurchaseOrderResponseResponse"/>

    </wsdl:operation>


    <wsdl:operation name="PushDespatchAdvice">

      <wsdl:input message="tns:PushDespatchAdviceInMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDespatchAdvice"/>

      <wsdl:output message="tns:PushDespatchAdviceOutMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDespatchAdviceResponse"/>

    </wsdl:operation>


    <wsdl:operation name="PushInvoice">

      <wsdl:input message="tns:PushInvoiceInMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushInvoice"/>

      <wsdl:output message="tns:PushInvoiceOutMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushInvoiceResponse"/>

    </wsdl:operation>


    <wsdl:operation name="PushDocument">

      <wsdl:input message="tns:PushDocumentInMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDocument"/>

      <wsdl:output message="tns:PushDocumentOutMsg"

        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDocumentResponse"/>

    </wsdl:operation>


  </wsdl:portType>


</wsdl:definitions>
```

C.6 SupplierPush.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="SupplierPush"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPush:1.0"

>


  <!-- ====== WSDL Import ====== -->


  <wsdl:import namespace="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0"

    location="SupplierPushInterface.wsdl"/>


  <!-- ====== Binding Definitions ====== -->


  <wsdl:binding name="SupplierPushSOAPBinding"
type="tns:SupplierPushPortType">

    <soap:binding style="document"

      transport="http://schemas.xmlsoap.org/soap/http"/>


    <wsdl:operation name="PushPurchaseOrderResponse">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushPurchaseOrderResponse"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>

    </wsdl:operation>


    <wsdl:operation name="PushDespatchAdvice">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDespatchAdvice"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>
```

```
    </wsdl:operation>

    <wsdl:operation name="PushInvoice">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushInvoice"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>

    <wsdl:operation name="PushDocument">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPush:1.0/PushDocument"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>

  </wsdl:binding>

  <!-- ====== Service Definitions ====== -->

  <wsdl:service name="SupplierPush">
    <wsdl:documentation>Web Service for Suppliers to invoke to transmit
      e-procurement documents using Push model</wsdl:documentation>
    <wsdl:port binding="tns:SupplierPushSOAPBinding"
name="SupplierPushPort">
      <soap:address location="JURISDICTION_BINDING_URL"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

C.7 SupplierPullInterface.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="SupplierPull"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0"

  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"

>



  <wsdl:types>

    <xsd:schema elementFormDefault="qualified"

      targetNamespace="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0"

      xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:
1.0"

      xmlns:nilns="urn:xml-gov-au:nehta:types:common:Nil:1.0"

      xmlns:queuens="urn:xml-gov-au:nehta:types:supply-chain:QueueDefs:
1.0"

      xmlns:xsd="http://www.w3.org/2001/XMLSchema">



      <!-- ====== Imports ====== -->

      <xsd:import namespace="urn:xml-gov-au:nehta:types:common:Nil:1.0"

        schemaLocation="../../types/common/Nil.xsd"/>

      <xsd:import namespace="urn:xml-gov-au:nehta:types:supply-
chain:QueueDefs:1.0"

        schemaLocation="../../types/supply-chain/QueueDefs.xsd"/>



      <!-- ====== Element Definitions ====== -->

      <xsd:element name="GetQueueStatus" type="nilns:Nil"/>

      <xsd:element name="GetQueueStatusResponse"
type="queuens:QueueStatus"/>

      <xsd:element name="PullPurchaseOrder" type="nilns:Nil"/>

      <xsd:element name="PullPurchaseOrderResponse"
type="queuens:OrderQueueHead"/>

      <xsd:element name="PullPurchaseOrderCancel" type="nilns:Nil"/>

      <xsd:element name="PullPurchaseOrderCancelResponse"

        type="queuens:OrderCancelQueueHead"/>

      <xsd:element name="PullPurchaseOrderChange" type="nilns:Nil"/>

      <xsd:element name="PullPurchaseOrderChangeResponse"

        type="queuens:OrderChangeQueueHead"/>

      <xsd:element name="PullRCTI" type="nilns:Nil"/>

      <xsd:element name="PullRCTIResponse"

        type="queuens:RCTIQueueHead"/>

      <xsd:element name="PullDocument" type="queuens:QueueName"/>
```

```
      <xsd:element name="PullDocumentResponse"
type="queuens:OtherQueueHead"/>
      <xsd:element name="UnknownQueueNameFault"
type="queuens:UnknownQueueName"/>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="GetQueueStatusInMsg">
    <wsdl:part name="body" element="tns:GetQueueStatus"/>
  </wsdl:message>

  <wsdl:message name="GetQueueStatusOutMsg">
    <wsdl:part name="body" element="tns:GetQueueStatusResponse"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderInMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrder"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderOutMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrderResponse"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderCancelInMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrderCancel"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderCancelOutMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrderCancelResponse"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderChangeInMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrderChange"/>
  </wsdl:message>

  <wsdl:message name="PullPurchaseOrderChangeOutMsg">
    <wsdl:part name="body" element="tns:PullPurchaseOrderChangeResponse"/>
  </wsdl:message>

  <wsdl:message name="PullRCTIInMsg">
    <wsdl:part name="body" element="tns:PullRCTI"/>
  </wsdl:message>

  <wsdl:message name="PullRCTIOutMsg">
    <wsdl:part name="body" element="tns:PullRCTIResponse"/>
  </wsdl:message>
```

```
  <wsdl:message name="PullDocumentInMsg">
    <wsdl:part name="body" element="tns:PullDocument"/>
  </wsdl:message>

  <wsdl:message name="PullDocumentOutMsg">
    <wsdl:part name="body" element="tns:PullDocumentResponse"/>
  </wsdl:message>

  <wsdl:message name="UnknownQueueNameFault">
    <wsdl:part name="body" element="tns:UnknownQueueNameFault"/>
  </wsdl:message>

  <wsdl:portType name="SupplierPullPortType">

    <wsdl:operation name="GetQueueStatus">
      <wsdl:input message="tns:GetQueueStatusInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0:GetQueueStatus"/>
      <wsdl:output message="tns:GetQueueStatusOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0:GetQueueStatusResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PullPurchaseOrder">
      <wsdl:input message="tns:PullPurchaseOrderInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrder"/>
      <wsdl:output message="tns:PullPurchaseOrderOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PullPurchaseOrderCancel">
      <wsdl:input message="tns:PullPurchaseOrderCancelInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderCancel"/>
      <wsdl:output message="tns:PullPurchaseOrderCancelOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderCancelResponse"/>
    </wsdl:operation>

    <wsdl:operation name="PullPurchaseOrderChange">
      <wsdl:input message="tns:PullPurchaseOrderChangeInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderChange"/>
      <wsdl:output message="tns:PullPurchaseOrderChangeOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderChangeResponse"/>
```

```
    </wsdl:operation>


    <wsdl:operation name="PullRCTI">
      <wsdl:input message="tns:PullRCTIInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullRCTI"/>
      <wsdl:output message="tns:PullRCTIOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullRCTIResponse"/>
    </wsdl:operation>


    <wsdl:operation name="PullDocument">
      <wsdl:input message="tns:PullDocumentInMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullDocument"/>
      <wsdl:output message="tns:PullDocumentOutMsg"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullDocumentResponse"/>
      <wsdl:fault message="tns:UnknownQueueNameFault"
        name="UnknownQueueNameFault"
        wsaw:Action="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/Fault/UnknownQueueNameFault"/>
    </wsdl:operation>


  </wsdl:portType>


</wsdl:definitions>
```

C.8 SupplierPull.wsdl

```
<?xml version="1.0"?>

<wsdl:definitions name="SupplierPull"

  targetNamespace="urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:
1.0"

  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

  xmlns:xsd="http://www.w3.org/2001/XMLSchema"

  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"

  xmlns:tns="urn:xml-gov-au:nehta:service:supply-chain:SupplierPull:1.0"

  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"

>


  <!-- ====== WSDL Import ====== -->


  <wsdl:import namespace="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0"

    location="SupplierPullInterface.wsdl"/>


  <!-- ====== Binding Definitions ====== -->


  <wsdl:binding name="SupplierPullSOAPBinding"
type="tns:SupplierPullPortType">

    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>


    <wsdl:operation name="GetQueueStatus">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0:GetQueueStatus"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>

    </wsdl:operation>


    <wsdl:operation name="PullPurchaseOrder">

      <soap:operation

        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrder"/>

      <wsdl:input>

        <soap:body use="literal"/>

      </wsdl:input>

      <wsdl:output>

        <soap:body use="literal"/>

      </wsdl:output>
```

```
    </wsdl:operation>


    <wsdl:operation name="PullPurchaseOrderCancel">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderCancel"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>


    <wsdl:operation name="PullPurchaseOrderChange">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullPurchaseOrderChange"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>


    <wsdl:operation name="PullRCTI">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullRCTI"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>


    <wsdl:operation name="PullDocument">
      <soap:operation
        soapAction="urn:xml-gov-au:nehta:service:supply-
chain:SupplierPull:1.0/PullDocument"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
```

```
      <wsdl:fault name="UnknownQueueNameFault">
        <soap:fault name="UnknownQueueNameFault" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>

  </wsdl:binding>

  <!-- ====== Service Definitions ====== -->

  <wsdl:service name="SupplierPull">
    <wsdl:documentation>Web Service for Suppliers to invoke to get e-
procurement
    documents using Pull model</wsdl:documentation>
    <wsdl:port binding="tns:SupplierPullSOAPBinding"
name="SupplierPullPort">
      <soap:address location="JURISDICTION_BINDING_URL"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```