Developer and Integration Guide

iPlanet Trustbase Payment Services

Version 2.0 Beta

July 2002
Rundate:
3:56, October 22, 2002

List of Figures

Contents

Introduction

The following chapter discusses all related documents to this guide.

Overall Layout

The complete documentation set for iPlanet Trustbase payment Services comprises of:

Installation Guide That explains how to install iPlanet Trustbase Payment Services and all its associated components.

System Administration Guide that explains how to configure and run iPlanet Trustbase Payment Services. It further illustrates how to make payments.

Developer and Integration Guide (this document) that provides a comprehensive example of how to deploy your own CPI application and how to integrate with your existing back end banking system. It a

The manual Covers:

- Integrating with your own backend by replacing Biab with your own Bank Back End application
- Installing and Running your CPI together with An example java source code on how to deploy a CPI application. Your application will then replace TooledUp.
- How to develop your own email acknowledgements

Detailed documentation can also be found on the iPlanet Trustbase Payment Services Website at

`http://docs.sun.com/?p=prod/s1.iptbpayment`

`http://docs.iplanet.com/docs/manuals/itps.html`

Related Documents

The following documents are considered pre-requisites to installing iPlanet Trustbase Payment Services (iTPS)

- Eleanor . iTPS is based on the Eleanor Technical Specification and as such you need to have familiarised yourself with this document.

  `http://www.identrus.com`

  Eleanor Scheme Technical Specification Version 1.0b

  Eleanor Scheme Operating Rules

  Eleanor Scheme Product Guide

  Note This Website requires a Username and password that should have been given to you when you joined the Identrus Scheme

- Identrus Message Specifications. iTPS is based on the IDentrus four corner model and as such four servers configures as identrus Transaction Coordinators (TC) using iPlanet Trustbase Transaction manager (iTTM) are assumed to be up and running. See `http://www.identrus.com`

  Identrus PKI Compliance (IT-PKI)

  Transaction Coordinator requirements (IT-TCFUNC)

  Identrus Smart Card Signing Interface Requirements (IT-SIR, ver 1.7)

  Core messaging specification (IT-TCMPD)

  Certificate Status Check Messaging specification (IT-TCCSC)

  Identrus Digital Signature Messaging System Specification (IT-DSMSSP, ver 2.0).

  Transaction Coordinator Certificate Status Check (CSC) Protocol Definition (IT-TCCSC, ver 2.0b)

  Note In order to access the documents within this website you need a Username and password that should have been given to you when you joined the Identrus Scheme.

- iPlanet Trustbase Transaction Manager (iTTM) documentation itself can be found below:

  http://docs.sun.com/?p=prod/s1.iptbtranm

  or in /cdrom/cdrom0

Software Requirements

Solaris(TM) 8 for SPARC(TM)

JDK 1.3.1

iPlanet Web Server 6.0 SP1

iPlanet Application Server 6.0 SP3

iPlanet Trustbase(TM) Transaction Manager 3.0.1

iMQ for Java 2.0

Oracle 8.1.7

Certificate Authority [e.g. iPlanet Certificate Management System 4.2]

Optional Hardware Security Module (HSM) on server [mandatory for Identrus participation - nCipher nShield 300 SCSI]

GemSAFE IS 1.1 for Identrus System 16000 Smartcards are configured on Buyer PC for use with Tooledup Seller Website.

Hardware requirements

Memory

Recommended single machine setup 512 MB

Disk Space

Recommended single machine setup 1 GB

Target Audience

System Administrators and Application Developers within the banking profession.

Chapter 1

Backend Integration

Introduction

Developing and integrating your own applications is covered in three main areas:

- Integrating with your own backend by replacing Biab with your own Bank Back End application. The is the subject of this chapter.
- Installing and Running your CPI together with An example java source code on how to deploy a CPI application. Your implementation of this example will then replace TooledUp. This is the subject of chapters 2 and 3.
- How to develop your own email acknowledgements. This is the subject of the final chapter 4.

Integration with Banks' Back End Systems

In order to achieve integration Banks are expected to replace Biab with the banks own back end system. iTPS is a consumer and producer of Eleanor XML messages that are pushed /read from from the Java Message Service Queue. Thus, the integration task is to get the XML message into a format that the banks back office can understand.

In order to understand existing ERA architecture you should consult the Eleanor Technical Specification, section 7, page 225 for details about what XML messaging has been adopted. There are two kinds of JMS Messaging models

1. Publish and Subscribe
   a. One producer can send a message to many consumers
   b. Every consumer receives a copy of the message sent by the producer
   c. The producer sending the messages is not dependent on the consumer receiving the messages
   d. This model could be broadly described as push-based model, where messages are broadcast to the consumers, without the consumers requesting or polling for new messages.
2. Point to Point
   a. This model allows clients to send and receive messages via channels known as queues.
   b. When a message is sent, only one receiver may consume that message.

The Transport mechanism implemented is iPlanet IMQ for Java 2.0, which is an implementation of the JMS messaging specification. It follows the Point-to-Point JMS model. You should consult the following document.

http://docs.sun.com/db/prod/s1.ipmsgquj

High Level Interaction Overview

When iTPS receives Identrus Eleanor messages it processes these messages, and translates the messages, into the xml messages specified in section 7 of the Eleanor Technical specification. These messages are then sent to the bank's back end systems via a queue. In the case of the ERA, we use the application called Bank-in-a-Box (Biab), to simulate a bank's systems, thus it is Biab that reads these messages off the queue specified in the ERA configuration. The users of Biab can view these messages, and send messages in response. These messages are sent back to iTPS via another specified queue. ITPS reads these messages off the queue, and translates these messages into Identrus Eleanor specified messages, and sends these messages to the appropriate parties.

Future Integration

Currently the application Biab is provided to simulate a bank's back end systems. This is only an example application for the ERA pilot. The banks' ultimate goal must be to replace this application and integrate this application's functionality into their banking systems. The exercise thus, is for the banks to implement an application that can:

1. Read the xml messages sent from iTPS from the specified queue.
2. Process these messages
3. Convert these messages into a format that can be understood by the bank's back end systems.
4. Process the information provided.
5. Convert the responses from the bank's back end system, into the xml messages defined in the Eleanor technical specification (section 7)
6. Put these xml messages on the specified queue for iTPS to read from, and process.
7. Plus, provide the ability to log and view these messages.

An example of how the back office processes incoming messages and provides responses to iTPS can be found in

/opt/itps-biab/src

Following a successful integration between iTPS and your back office, this guide now discusses how customers can send Eleanor payment messages in a similar way to the tooled up demo website.

Chapter 2

Customer Integration

Overview

This chapter illustrates a typical payment being processed and describes how the payment can be initiated using the iPlanet Trustbase Payment Services Corporate Payment Initiation Library API (CPI). There are three kinds of situations that may warrant the use of the CPI library. Your own implementation of this will then replace Tooledup.

Figure 2-1 Buyer buys something from Sellers Website

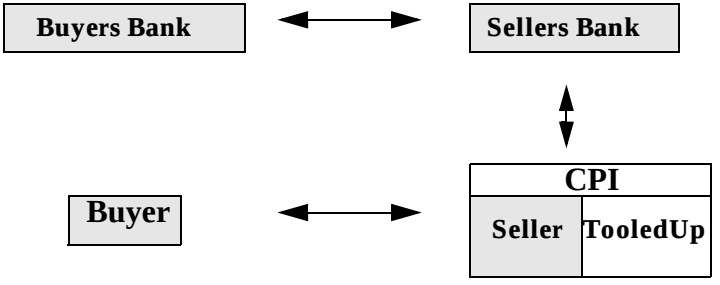


Figure 2-2 Buyer is making a payment with a Sellers signature

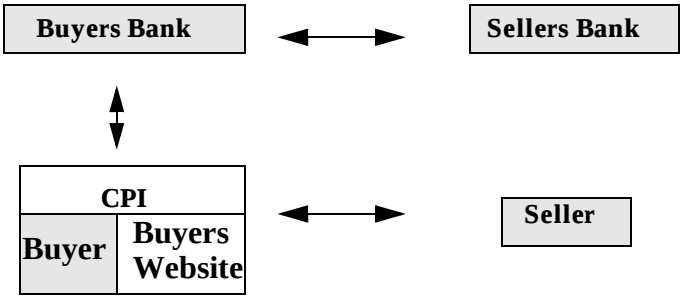
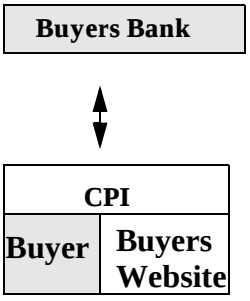


Figure 2-3 Buyer trying to make a payment without the need for the seller's signature



Using the iTPS API to initiate a Payment

The Corporate Payment Initiation Library (CPI) allows a merchant (seller) web site or another application (such as a bank web site) to submit payment requests to a participant (such as the seller / buyer bank).

In order to submit a payment, typically a merchant performs the following steps:

1. Collects payment data from a customer (e.g. from HTML forms submitted to the merchant's web site).
2. Creates an XML element that represents the payment.
3. Sends the XML to the customer for signing
4. Creates any additional XML elements as required (e.g. SellerPrivateData, RemittanceData).
5. Creates a ConfigAdapter. The config adapter allows the merchant application to specify the location (URL) of his financial institution, the certificate to be used for signing payment messages and initialises the SSL subsystem.
6. Optionally defines the transport over which the message is sent using TransportAdapter
7. Submits the XML, signed data and ConfigAdapter to the PaymentInitiator which will construct a signed payment request and send it to the merchant's financial institution.

8. Receives from the PaymentInitiator a Status object, which contains the XML reply content and raw data suitable for logging.
9. Parses the XML reply content to establish the status of the payment request using other API methods e.g. responseReceived()

NOTE Consult your iTPS and iTTM API for more information on this:

com.iplanet.trustbase.initiator

com.iplanet.trustbase.initiator.cpi

com.iplanet.trustbase.initiator.config

com.iplanet.trustbase.initiator.transport

The above iTPS API can be found within the docs directory on your CD-ROM drive.

/cdrom/cdrom0/opt/itps-cpi/javadocs

The Trustbase API can be found within the iTTM installation directory or from your CD-ROM drive

Parameters needed to send a message

In order to send a message the following parameters need to be defined

1. role - identifies the role of the customer (for example ‘BU:01’ = Buyer, ‘SE:01’= Seller)
2. doctype - the doctype of the request, for example

   ```
   <!DOCTYPE PaymentRequest PUBLIC \"-//IDENTRUS//ELEANOR PAYMENT
   REQUEST DTD//en"
   "http://www.identrus.com/Eleanor/1.0/ver1.0/PaymentRequest.dtd">
   ```

3. data - the data signed by the signature(s) in the pkcs7 parameter
4. elements - additional XML elements that belong in the request (e.g. Header, RemittanceData).
5. pkcs7 - array of the base64 encoded pkcs7 signed object data, each of which represents the buyer’s signature. If multiple signatures are given, the multipleSignaturesType needs to be one of

   ALL_SIGN_SAME_DATA, SIGN_PRECEDING_SIGNATURES.

   If the pkcs7 array parameter contains one signature, the multipleSignaturesType parameter is ignored. Multiple signatures needs to be either: all sign the same application data (but not sign preceding signatures) or each (except the first) sign the preceding signatures, forming a chain.
6. config - the com.iplanet.trustbase.initiator.config.ConfigAdapterImpl for this operation the following properties needs to be present:
   a. Keystore Domain Space
   b. Signing Certificate
   c. Verification Certificate
   d. Optionally an SSL signing certificate
7. multipleSignaturesType - describes what type the multiple signatures are in the pkcs7 array parameter. This will be one of
   a. ALL_SIGN_SAME_DATA
   b. SIGN_PRECEDING_SIGNATURES.
8. transport - the com.iplanet.trustbase.initiator.TransportAdapter for this operation. This is Optional

Certificate Verification

The CPI packages behavior can be altered by setting properties in the ConfigAdapter. The recognised properties are defined in

`com.iplanet.trustbase.initiator.PropertyCodes`

This package contains details of how the Property codes can be used

How to use the API

The following sample code illustrates how to use the API and illustrate initiating a payment and sending it to iPlanet Trustbase Payment Services. There are two main aspects to this:

- How to build the CPI Library parameters
- How to send a Payment Initiation Message

```
class PaymentInitiatorExample
{
     public Status processPaymentRequest()
     {

// Set up the parameters required to send a PaymentRequest
// The role. In this case we are acting as the seller, hence the role is "SE:01"
          String role ="SE:01";

// The XML DOCTYPE for the message to be sent. The doctype identifies the type of the message,
// and the DTD that defines the structure of the message. In this case we are sending a payment
// request, for which the public ID is -//ELEANOR PAYMENT REQUEST DTD//en and the System ID
// is http://www.identrus.com/Eleanor/1.0/ver1.0/PaymentRequest.dtd.
          String doctype = "<!DOCTYPE PaymentRequest PUBLIC \"-//IDENTRUS//ELEANOR PAYMENT REQUEST
DTD//en\""
                         +" \"http://www.identrus.com/Eleanor/1.0/ver1.0/PaymentRequest.dtd\"> ";

// Get the data describing the payment. This is the data that will be signed by the customer
          String customerData = createCustomerData(getPaymentAmount());

// Get the other elements to be included in the request, such as the Header block.
          String[] elements = getElements();

// Request that the customer signs the payment request. This signature is performed by the
// smartcard plugin in the customer's browser.
          String[] customerSig = getCustomerSignature(customerData);

// The signature type we are using. This defines how signatures are applied if there
// are multiple signatures in the customerSig array. In this example code there is a single signature
// from the customer, so the parameter is not used regardless of the value being set to
// PaymentInitiator.ALL_SIGN_SAME_DATA as per below
          int multipleSignatureType = PaymentInitiator.ALL_SIGN_SAME_DATA;

// Get the config adapter for this request. The config adapter is the means by which the
// PaymentInitiator class accesses the underlying PKI and properties of the system.
          ConfigAdapter config = getConfig();

// The transport adapter defines the transport over which the message is sent. Here we
// will use HTTP
          TransportAdapter transport = new HTTPTransportAdapter();

// Get an instance of the Payment Initiator,
```

```
          PaymentInitiator pi= PaymentInitiatorManager.getPaymentInitiator();


// Send the request to the Payments server using the information gathered above.
          Status status = pi.send(role,
                                        doctype,
                                        customerData,
                                        elements,
                                        customerSig,
                                        multipleSignatureType,
                                        config,
                                        transport);

// Retrieve the response data from the Payments Server check to see if the response is true or false
          if (status.responseReceived())
          {
// Retrive the actual content as a String object
               String response = status.getContent();

               // Business logic to deal with the response
               // ...
               // ...
               // ...
          }
          else if   (status.hasConnectionFailed())
          {
               // No response - log an error
          }
          else
          {
               // fall through error handling
          }
     }


     public ConfigAdapter getConfig()throws Exception
     {
// The default implementation of ConfigAdapter works from information supplied in a Properties
// object. Certificates are retrieved from a file on disk.

          Properties props = new Properties();

props.put(PropertyCodes.INITIATOR_KEYSTORE_DOMAIN_SPACE,"mykeystore");
props.put(PropertyCodes.INITIATOR_KEYSTORE_SIGNING_CERTIFICATE,"server-cert);
props.put(PropertyCodes.INITIATOR_KEYSTORE_VERIFICATION_CERTIFICATE + ".1","IdentrusRoot");
return new ConfigAdapterImpl(props);
     }

     public String createCustomerData(String paymentAmount)
     {
// Construct the CustomerSignedData XML block for the customer to sign.
// There are a number of ways of constructing this block, including the
```

```
// iPlanet JAXHIT technology.  There are a number of advantages to using
// this technology from iPlanet, such as type checking and XML validity checking.
// For brevity we will just create the block as a String in this example.
// The values here are hard coded for example only and would typically be
// supplied by the calling Seller application

          // The ISO currency specified in the Eleanor Tech Spec, here we're using US $
          String currency = "USD";

          // The valueDate is taken from the transaction attributes agreed between Buyer and
          // Seller
          String valueDate = "2001-07-25";

          // The unique Eleanor ID for the request as per the Eleanor Tech Spec
          String eleanorRef = this.getEleanorRef();

          // The name of the seller company (i.e. us)
          String sellerName = "ACME Soap Co";

          // The type of obligation. In this case it is "NONE", as we are sending a plain revocable
          // payment request
          String obligation = "NONE";

          String customerData = "<SubscriberSignedData><NegotiatedData><Amt>"
                                      +paymentAmount
                                      +"</Amt><CurCode>"
                                      +currency+
                                      "</CurCode><RequestedValueDate>"
                                      +valueDate+

"</RequestedValueDate></NegotiatedData><SellerPublicData><EleanorTransactionReference>"
                                      +eleanorRef+
                                      "</EleanorTransactionReference><SellerName>"
                                      +sellerName
                                      +"</SellerName></SellerPublicData><Obligation>"
                                      +obligation+
                                      "</Obligation></SubscriberSignedData>";
          return customerData;

     }

     public String getElements()
     {
          // Construct the header block for the message
          // Eleanor product code as defined in the Eleanor Specification. "xPx" is the code
          // for Payment Request
             String product = "xPx";

          // The XML doc type (the type of the root element of the message)
             String docType = "PaymentRequest";

          // The version
```

```
            String version = "1.0";

         // Build the header block
            return new String[]{"<Header xml:lang=\"enGB\"><Product>"
                                    +Product
                                    +"</Product><DocumentType>"
                                    +DocType
                                    +"</DocumentType><Version>"
                                    +Version
                                    +"</Version></Header>"};
     }


     private String getEleanorRef()
     {
// return a unique Eleanor reference number as per the Eleanor Tech Spec
     }

     public String getPaymentAmount()
     {
// Get the payment amount attribute from the transaction
     }

     public String[] getCustomerSignature(String customerData)
     {
          // Request that the customer signs the customerData with their smartcard.
          // ...
          // ...
          // ...
     }
}
```

Test.java

The full source for this worked example can be found in:

```
/cdrom/cdrom0/iTPS/cpi/cpi.tar
```

The script to run test.java can be found in:

```
/opt/iTPS-CPI/bin/test.sh
```

The source can be found in:

```
/opt/iTPS-CPI/example/com/example/example1/Test.java
```

Chapter 3

Installing your CPI

The followibng chapter illustrates how to install and run an example that shows how to deploy the CPI.

Installing the CPI API

1. TThe CPI component contains several directories and files that are detailed below:
   - /opt/itps-cpi/bin : contains scripts that will set your classpath and help you run the tools you will need. The scripts are all written for use with bourne shell.
   - /opt/itps-cpi/lib : contains all the binaries that the CPI will need to run – this includes shared objects and jarfiles.
   - /opt/itps-cpi/store : This directory will be used to store your TokenKeyStore.
   - /opt/itps-cpi/doc : API documentation and TokenKeyTool detailed documentation.
   - /opt/itps-cpi/example: Source Code.
2. It does not matter whether iTTM and iTPS are running during installation. However they, and all their associated components such as iAS and iWS, should be running if you need to run this component
3. An external JDK 1.3.1 needs to be downloaded either from http://www.javasoft.com to /usr/java1.3
4. You are now ready to install CPI. From the root, run the UNIX install script and answer the questions. For example

   ```
   /cdrom/cdrom0/itps-cpi/setup -c
   ```

   This will automatically install to

   ```
   /opt/itps-cpi
   ```
5. You are now required to use TokenKeyTool. A description of this can be found, either within your iTTM manual or in

   /opt/itps-cpi/doc/TokenKeyTool.html

   By typing help when running TokenKeyTool you can obtain details of how this should be used. To run this script type:

   ```
   cd /opt/itps-cpi/bin/
   ./tok.sh
   ```
6. Before you can proceed you will need some trusted certificates. These certificates are in files that you have access to and each of the certificate files contain a single PEM format certificate. The certificates required are.

- C1 : The IRCA certificate (In the example below this is called e.g. IRCA.crt) This is referred to as the verification certificate.
- C2 : The L1CA Certificate.(In the example below this is called L1CA.crt)

Finally you will need to issue a request for a Seller signing certificate and Buyer signing certificate, then import the generated Certificate Chains into your CertStore.

- C3 : The Seller's Signing Certificate e.g. Seller SC
- C4 : The Buyer's Signing Certificate e.g. Buyer SC

7. The following distinguished names should be used:

C1: "IRCA" certificate e.g. "CN=Identrus Root,OU=Identrus Root,O=Identrus,C=US"

C2: "L1CA" certificate e.g. "CN=L1 Bank CA,OU=L1 Bank,O=L1,C=GB"

C3: "SellerSC" certificate e.g. "CN=Seller SC, OU=L1 Bank, O=L1"

C4: "BuyerSC" certificate e.g. "CN=Buyer SC, OU=L1 Bank, O=L1"

8. In order to create your store the following steps need to be performed:

a. Run the tok.sh script that starts the tokenkeytool.

b. Type help to obtain details of useage

c. Create A Trust Domain using openstoremanager command eg openstoremanager –domainspace "file:///opt/itps-cpi/store" –manager local.

d. Create a TokenKeyStore using the createstore command e.g. createstore –store identrus (you will be prompted to give a password - please record this password).

e. Import your trusted IRCA (C1) and L1CA (C2) Certificates files using the command importtrustedcerts, e.g.:

importtrustedcerts –file "IRCA.crt"

importtrustedcerts –file "L1CA.crt"

f. Generate a holding key pair for your Seller and Buyer Signing Certificates using the command genkey, e.g.:

genkey –dname "CN=SellerSC,OU=L1 Bank,O=L1"

genkey –dname "CN=BuyerSC,OU=L1 Bank,O=L1"

g. View the keys to acquire the generated aliases for them using the command listkeys e.g. listkeys.

h. Generate PKCS7 certificate requests to be processed by your L1CA using the command certreq, e.g.:

certreq –alias <generated_key_alias> –dname “CN=SellerSC,OU=L1 Bank,O=L1” –file “sellercertrequest”

certreq –alias <generated_key_alias> –dname “CN=BuyerSC,OU=L1 Bank,O=L1” –file “buyercertrequest”

i. Paste the generated certificate requests into your L1CA and get the CA generated Base64 encoded certificate chains. Store them in files called “sellercertresponse” and “buyercertresponse”

j. Import the certificates into the certificate store using the command importkeychain, e.g.:

importkeychain –file “sellercertresponse”

importkeychain –file “buyercertresponse”

k. Quit the TokenKeyTool using the command quit.

9. We now illustrate this with an example

```
Script started on Mon 24 Sep 2001 17:01:34 BST
myhost# cd /opt/itps-cpi/bin/
myhost#./tok.sh
TokenKeyTool> openstoremanager -domainspace
"file:///opt/itps-cpi/store" -manager local
TokenKeyTool> createstore -store identrus
Login to JSS token Internal Key Storage Token: password
TokenKeyTool> importtrustedcerts -file "IRCA.crt"
TokenKeyTool> importtrustedcerts -file "L1CA.crt"
TokenKeyTool> genkey -dname "CN=SellerSC,OU=L1 Bank,O=L1”
TokenKeyTool> genkey -dname "CN=BuyerSC,OU=L1 Bank,O=L1”
TokenKeyTool> listkeys
+KeyEntrys
  +KeyEntry
    subject name: CN=SellerSC,OU=L1 Bank,O=L1
```

```
    issuer name: CN=SellerSC,OU=L1 Bank,O=L1
    serial #: 0x7733ad362cc3ecce
    +aliases
      alias: 7733ad362cc3ecce#CN=SellerSC,OU=L1 Bank,O=L1
      +certificate chain
        +certificate [0]
        subjectName: CN=SellerSC,OU=L1 Bank,O=L1
        issuerName: CN=SellerSC,OU=L1 Bank,O=L1
        serial#: 0x7733ad362cc3ecce
        not before: 24-Sep-01 16:03:20
        not after: 24-Sep-02 16:03:20
+KeyEntry
    subject name: CN=BuyerSC,OU=L1 Bank,O=L1
    issuer name: CN=BuyerSC,OU=L1 Bank,O=L1
    serial #: 0x7733ad362cc3eccf
    +aliases
      alias: 7733ad362cc3eccf#CN=BuyerSC,OU=L1 Bank,O=L1
      +certificate chain
        +certificate [0]
        subjectName: CN=BuyerSC,OU=L1 Bank,O=L1
        issuerName: CN=BuyerSC,OU=L1 Bank,O=L1
        serial#: 0x7733ad362cc3eccf
        not before: 24-Sep-01 16:03:20
        not after: 24-Sep-02 16:03:20

TokenKeyTool> certreq -alias "7733ad362cc3ecce#CN=SellerSC,OU=L1
Bank,O=L1" -file "sellercertrequest"
TokenKeyTool> certreq -alias "7733ad362cc3eccf#CN=BuyerSC,OU=L1
Bank,O=L1" -file "buyercertrequest"
TokenKeyTool> importkeychain -file "sellercertresponse"
+KeyEntry
```

```
  subject name: CN=SellerSC,OU=L1 Bank,O=L1
  issuer name: CN=L1CA,OU=L1 Bank,O=L1,C=GB
  serial #: 0x10a
  +aliases
    alias: 10a#CN=L1CA,OU=L1 Bank,O=L1,C=GB
    +certificate chain
      +certificate [0]
        subjectName: CN=SellerSC,OU=L1 Bank,O=L1
        issuerName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        serial#: 0x10a
        not before: 24-Sep-01 16:09:23
        not after: 19-Sep-02 08:23:24
      +certificate [1]
        subjectName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x18
        not before: 19-Sep-01 08:23:24
        not after: 19-Sep-02 08:23:24
      +certificate [2]
subjectName: CN=Identrus Root,OU=Identrus Root,O=Identrus,C=US
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x1
        not before: 29-Aug-01 00:00:00
        not after: 29-Aug-03 00:00:00
TokenKeyTool> importkeychain -file "buyercertresponse"
+KeyEntry
  subject name: CN=BuyerSC,OU=L1 Bank,O=L1
  issuer name: CN=L1CA,OU=L1 Bank,O=L1,C=GB
  serial #: 0x10b
```

```
  +aliases
    alias: 10b#CN=L1CA,OU=L1 Bank,O=L1,C=GB
    +certificate chain
      +certificate [0]
        subjectName: CN=BuyerSC,OU=L1 Bank,O=L1
        issuerName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        serial#: 0x10b
        not before: 24-Sep-01 16:09:23
        not after: 19-Sep-02 08:23:24
      +certificate [1]
        subjectName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x18
        not before: 19-Sep-01 08:23:24
        not after: 19-Sep-02 08:23:24
      +certificate [2]
subjectName: CN=Identrus Root,OU=Identrus Root,O=Identrus,C=US
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x1
        not before: 29-Aug-01 00:00:00
        not after: 29-Aug-03 00:00:00

TokenKeyTool> listkeys
+KeyEntrys
  +KeyEntry
    subject name: CN=SellerSC,OU=L1 Bank,O=L1
    issuer name: CN=L1CA,OU=L1 Bank,O=L1,C=GB
    serial #: 0x10a
    +aliases
```

```
      alias: 10a#CN=L1CA,OU=L1 Bank,O=L1,C=GB
      +certificate chain
        +certificate [0]
        subjectName: CN=SellerSC,OU=L1 Bank,O=L1
        issuerName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        serial#: 0x10a
        not before: 24-Sep-01 16:09:23
        not after: 19-Sep-02 08:23:24
        +certificate [1]
        subjectName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x18
        not before: 19-Sep-01 08:23:24
        not after: 19-Sep-02 08:23:24
        +certificate [2]
        subjectName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x1
        not before: 29-Aug-01 00:00:00
        not after: 29-Aug-03 00:00:00
+KeyEntry
    subject name: CN=BuyerSC,OU=L1 Bank,O=L1
    issuer name: CN=L1CA,OU=L1 Bank,O=L1,C=GB
    serial #: 0x10b
    +aliases
      alias: 10a#CN=L1CA,OU=L1 Bank,O=L1,C=GB
      +certificate chain
        +certificate [0]
        subjectName: CN=BuyerSC,OU=L1 Bank,O=L1
```

```
        issuerName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        serial#: 0x10b
        not before: 24-Sep-01 16:09:23
        not after: 19-Sep-02 08:23:24
        +certificate [1]
        subjectName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x18
        not before: 19-Sep-01 08:23:24
        not after: 19-Sep-02 08:23:24
        +certificate [2]
        subjectName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
        serial#: 0x1
        not before: 29-Aug-01 00:00:00
        not after: 29-Aug-03 00:00:00

TokenKeyTool> listcerts
+TrustedCertificateEntrys
  +TrustedCertificateEntry
    +aliases
      alias: 1#CN=Identrus Root,OU=Identrus Root,O=Identrus,C=US
    +certificate
      subjectName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
      issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
      serial#: 0x1
      not before: 29-Aug-01 00:00:00
```

```
      not after: 29-Aug-03 00:00:00
  +TrustedCertificateEntry
    +aliases
      alias: 18#CN=Identrus Root,OU=Identrus Root,O=Identrus,C=US
    +certificate
      subjectName: CN=L1CA,OU=L1 Bank,O=L1,C=GB
      issuerName: CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US
      serial#: 0x18
      not before: 19-Sep-01 08:23:24
      not after: 19-Sep-02 08:23:24
TokenKeyTool> quit
myhost# exit
myhost#
script done on Mon 24 Sep 2001 17:12:28 BST
```

10. Now you are ready to run the test harness – you must alter the script called test.sh which can be found in the directory:

/opt/itps-cpi/bin

The test.sh script has defaults for all paramters needed. The parameters it expects are as follows.

a. Payment amount.

b. Payment currency

c. Payment date.

d. Payment account

e. Payment reference

f. Keystore domainspace+store eg file:/ / /opt/itps-cpi/store#identrus

g. Keystore password

h. Verification certificate alias (i.e. {IRCAserial number}#{issuerDN})

i. Seller signing certificate alias (i.e. {SellerSC serial number}#{issuerDN})

j. Buyer signing certificate alias (i.e. {BuyerSC serial number}#{issuerDN})

11. In order to see the certificate aliases, type the following preparatory commands

```
cd /opt/itps-cpi/bin/
```

```
./tok.sh
```

```
openstoremanager -domainspace "file:///opt/itps-cpi/store"
-manager local
```

```
setdefaultstore -manager local -store identrus
```

a. For the verification certificates, type

```
listcerts
```

b. For the Signing Certificates, type

```
listkeys
```

12. Run the test program and receive a response from your TC. Before running the test script make sure jmqbroker, jmsproxy, slapd, iAS, iWS, iTTM, iTPS and Biab backend are all running or a Doctype error will occur. If the status field in the response message =”success” then the test is successful. It looks something like the example below.

```
Script started on Mon 24 Sep 2001 17:30:38 BST

ragnarok# ./test.sh

Init Seller [ password ] [ file:///opt/itps-cpi/store#identrus ] [ 10a#CN=L1CA,OU=L1
Bank,O=L1,C=GB] [ 1#CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US]

Init Buyer [ password ] [ file:///opt/itps-cpi/store#identrus ] [ 10b#CN=L1CA,OU=L1
Bank,O=L1,C=GB] [ 1#CN=Identrus Root,OU=Identrus
Root,O=Identrus,C=US]

**********

CN=L1CA;

OU=L1 Bank;

O=L1;

C=GB

**********

CN=L1CA;

OU=L1 Bank;

O=L1;

C=GB

**********

CN=L1CA;

OU=L1 Bank;

O=L1;

C=GB

**********

CN=L1CA;

OU=L1 Bank;

O=L1;

C=GB

    ----------------------------------------------------------------
```

```
RESPONSE BEGIN

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE Acknowledgement
PUBLIC "-//IDENTRUS//ELEANOR ACKNOWLEDGEMENT DTD//en"
"file:///bankInterface.dtd"><Acknowledgement><NIB
id="NIB_9F6B3D9AFBDEEAA7AB9C8DC84EB532DAB6606008_1"><ContextInfo
msggrpid="32C2BC35A480C2D8F1FA34AFA4E34979D211504B"
msgid="SFI01"/><StartTime><LocalTime
id="LocalTime_9F6B3D9AFBDEEAA7AB9C8DC84EB532DAB6606008_1"
time="20020522135601Z"/></StartTime><MsgTime><LocalTime
id="LocalTime_9F6B3D9AFBDEEAA7AB9C8DC84EB532DAB6606008_2"
time="20020522135605Z"/></MsgTime></NIB><Signature><SignedInfo><Can
onicalizationMethod
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/><SignatureMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#rsa-sha1"/><Reference
URI="#NIB_9F6B3D9AFBDEEAA7AB9C8DC84EB532DAB6606008_1"><Transforms><
Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"/><DigestValue>ja
8Ls8YcPn/jNDTxaeMRf0aKUaA=</DigestValue></Reference><Reference
URI="#ContentAcknowledgement_623FD0E43DD72CAB72DCC1F79C94A7746A6AF5
42_1"><Transforms><Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"/><DigestValue>MJ
2Ls4HHn3QeuntpjvEYMUksdzw=</DigestValue></Reference><Reference
URI="#Response_623FD0E43DD72CAB72DCC1F79C94A7746A6AF542_1"><Transfo
rms><Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"/><DigestValue>Jh
AXLM+2OKrVQCeNoUg23mO/UaM=</DigestValue></Reference></SignedInfo><S
ignatureValue>AHhfU3zHMEVWl9ZB9hu6xW8UN7Go81j30LOXG+dhDM9CpnDUtjEXP
CDufOYHGtLu

zupK1g2k0llFQG9wv1PmIXCasIPaZDFhsxjE16msfH8KjTdqDXWh9YuK5r0A90J2

84fgTGQ0tOKbfklWKIF2kgjsQTL/xm+b09FFVpUl8Dw=</SignatureValue><KeyIn
fo><X509Data><X509IssuerSerial><X509IssuerName>CN=Identrus Root
Certificate Manager,OU=Identrus
Root,O=Identrus,C=US</X509IssuerName><X509SerialNumber>266</X509Ser
ialNumber></X509IssuerSerial></X509Data></KeyInfo></Signature><Cert
Bundle><X509Data><X509IssuerSerial><X509IssuerName>CN=Identrus Root
Certificate Manager,OU=Identrus
Root,O=Identrus,C=US</X509IssuerName><X509SerialNumber>266</X509Ser
ialNumber></X509IssuerSerial><X509Certificate>MIIDYzCCAkugAwIBAgIBG
TANBgkqhkiG9w0BAQQFADBkMQswCQYDVQQGEwJVUzER
```

```
.....................................................


GDFDpV46exYYgbclRMs37kyLPn/tARFeknc09aXEbZrYaMRUy0Q5kfCb71F2</X509C
ertificate></X509Data></CertBundle><ContentAcknowledgement
id="ContentAcknowledgement_623FD0E43DD72CAB72DCC1F79C94A7746A6AF542
_1"><Header
xml:lang="EN"><Product>xPx</Product><DocumentType>Acknowledgement</
DocumentType><Version>1.1</Version></Header><References><EleanorTra
nsactionReference>73aa07ed-f876-6d18-8000-2120448280a1</EleanorTran
sactionReference></References><AcknowledgementData><Acknowledgement
Type>PayInst</AcknowledgementType><Status>SUCCESS</Status><R
easonCode>00SP00</ReasonCode><ReasonText>Message
validated</ReasonText></AcknowledgementData></ContentAcknowledgemen
t><Response
id="Response_623FD0E43DD72CAB72DCC1F79C94A7746A6AF542_1"><ResponseD
ata>MIIIiAoBAKCCCIEwggh9BgkrBgEFBQcwAQEEgghuMIIIajCBpqFAMD4xEDAOBgN
V

.....................................................


iz5/7QERXpJ3NPWlxG2a2GjEVMtEOZHwm+9Rdg==</ResponseData><CSCResponse
><NIB
id="NIB_917C8CEF242EA78F011F593F342981DA537F7673_1"><ContextInfo
msggrpid="AA6DEEF8DCBA3322287AF42D55C1DC711C91FF72"
msgid="1022065168495"/><StartTime><LocalTime
id="LocalTime_917C8CEF242EA78F011F593F342981DA537F7673_1"
time="20020522105903Z"/></StartTime><MsgTime><LocalTime
id="LocalTime_917C8CEF242EA78F011F593F342981DA537F7673_2"
time="20020522105928Z"/></MsgTime></NIB><Signature><SignedInfo><Can
onicalizationMethod
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/><SignatureMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#rsa-sha1"/><Reference
URI="#NIB_917C8CEF242EA78F011F593F342981DA537F7673_1"><Transforms><
Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/></Transforms><DigestMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"/><DigestValue>Og
eKIIySzLyZuaCfS3bcBsRTiMQ=</DigestValue></Reference><Reference
URI="#Response_353017FFA82146F17316CEBFA48B9EC12C974186_1"><Transfo
rms><Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-
hiroshi-dom-hash-03.txt"/></Transforms><DigestMethod
```

```
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"/><DigestValue>i+
UtlGeGfWYNAJeM+b3VrEYneRQ=</DigestValue></Reference></SignedInfo><S
ignatureValue>J1/h3RfMhG6NYtn6UKvx168DWy1WBQGaM1PSPUMK8Ap94difqX3Dz
wDk5bibr7wI

WjJt9D/62wOlIX4ZGDQXMdJ1CWW5Wlq/CiotH5VdHjnp7709fbI8CTmMOQEaZ7TI

tBMVYxjLK8HvosJuy4lPesu88hMuY4PbztuKQk0k18k=</SignatureValue><KeyIn
fo><X509Data><X509IssuerSerial><X509IssuerName>CN=Identrus Root
Certificate Manager,OU=Identrus
Root,O=Identrus,C=US</X509IssuerName><X509SerialNumber>7</X509Seria
lNumber></X509IssuerSerial></X509Data></KeyInfo></Signature><CertBu
ndle><X509Data><X509IssuerSerial><X509IssuerName>CN=Identrus Root
Certificate Manager,OU=Identrus
Root,O=Identrus,C=US</X509IssuerName><X509SerialNumber>7</X509Seria
lNumber></X509IssuerSerial><X509Certificate>MIIDajCCAlKgAwIBAgIBBzA
NBgkqhkiG9w0BAQQFADBkMQswCQYDVQQGEwJVUzER

.............................................................


GDFDpV46exYYgbclRMs37kyLPn/tARFeknc09aXEbZrYaMRUy0Q5kfCb71F2</X509C
ertificate></X509Data></CertBundle><Response
id="Response_353017FFA82146F17316CEBFA48B9EC12C974186_1"><ResponseD
ata>MIIIiAoBAKCCCIEwggh9BgkrBgEFBQcwAQEEgghuMIIIajCBpqFAMD4xEDAOBgN
V

.............................................................


cyNUjoMo+uJXI4S4WcaflNUpadLeTat7FKLgtbVpbRgxQ6VeOnsWGIG3JUTLN+5M

iz5/7QERXpJ3NPWlxG2a2GjEVMtEOZHwm+9Rdg==</ResponseData></Response><
/CSCResponse></Response></Acknowledgement>

RESPONSE END

------------------------------------------------------------------

myhost# exit

myhost#

script done on Mon 24 Sep 2001 17:31:20 BST
```

Sending Payment Message to host other than AIA in the Certificate

It is possible to override the default destination of the payment message used in the Test.java example. This is done by adding two properties to the ConfigAdapter for the Seller in getConfigSeller() as follows:

```
props.put ( PropertyCodes.INITIATOR_LOCATION_FORCE_DEFAULT,
"true" );

props.put ( PropertyCodes.INITIATOR_LOCATION_DEFAULT,
"http://{target hostname}/NASApp/NASAdapter/TbaseNASAdapter" );
```

Now recompile the example, e.g.

```
cd /opt/itps-cpi/bin

. ./cp.sh

javac ../example/com/example/example1.Test.java
```

Then you MUST add the new classfile into the classpath by adding the following line to cp.sh (Just before the export statements):

```
CLASSPATH=/opt/itps-cpi/example:$CLASSPATH
```

Chapter 4

Creating Communicative Templates

Introduction

This help sheet describes the processing that takes place for presentment of XML documents relating to email acknowledgements and Biab Backend Messages. iTTM uses the XSLT translation technology see

http://www.w3.org/TR/xslt

from the Apache open source project found at http://xml.apache.org/. iTTM uses the Xalan XML parser for Java with the Xerces XML parser. See

http://xml.apache.org/xalan-j/index.html

http://xml.apache.org/xerces-j/

http://xml.apache.org/xerces2-j/javadocs/xerces2/overview-summary.html

We now discuss under the following main headings:

- Email acknowledgements
- Biab BackEnd Messages

Email Acknowledgements

Method

iTPS uses two stylesheets to transform the Eleanor XML acknowledgement into the email that is sent to the subscriber. These stylesheets are found at

```
/opt/ittm/current/Config/Templates/en/Payment/subject.xsl
```

that creates the email subject and

```
/opt/ittm/current/Config/Templates/en/Payment/body.xsl
```

creates the body.

Standard Output response

Converting body.xsl produces an html output as follows

```
<html>
   <head>
      <title/>
   </head>
   <body>
      <p>
         <table>
            <tr>
               <td>
                  <i>Transaction Reference:</i>
               </td>
               <td>1766631e9fdc51f1180002120448330</td>
            </tr>
            <tr>
               <td>
                  <i>Status:</i>
               </td>
               <td>SUCCESS</td>
            </tr>
            <tr>
               <td>
                  <i>Reason Text:</i>
               </td>
               <td>Payment Instruction Details Accepted</td>
            </tr>
         </table>
      </p>
   </body>
</html>
```

Changing the Output

The following documents provide more information about XSLT

`http://www.w3.org/Style/XSL/#learn`

`http://www.nwalsh.com/docs/tutorials/xsl/xsl/frames.html`

In order to change the acknowledgement email messages you should adopt the following procedure.

1. Download XML parsing software as follows

   a. Either Instant Saxon 6.2.2 command line XSL parser for Microsoft Windows.

   `http://users.iclway.co.uk/mhkay/saxon/`

   b. Or alternatively there is a Java packaged version.

   `http://xml.apache.org/crimson/index.html`

2. The following illustrates how to run the Saxon tool. If no parameters are specified then the following output is provided:

```
c:\temp>saxon
No source file name
SAXON 6.2.2 from Michael Kay
Usage: saxon [options] source-doc style-doc {param=value}...
Options:
  -a              Use xml-stylesheet PI, not style-doc argument
  -ds             Use standard tree data structure
  -dt             Use tinytree data structure (default)
  -o filename     Send output to named file or directory
  -m classname    Use specified Emitter class for xsl:message
output
  -r classname    Use specified URIResolver class
  -t              Display version and timing information
  -T              Set standard TraceListener
  -TL classname   Set a specific TraceListener
  -u              Names are URLs not filenames
  -w0             Recover silently from recoverable errors
  -w1           Report recoverable errors and continue (default)
  -w2             Treat recoverable errors as fatal
  -x classname    Use specified SAX parser for source file
  -y classname    Use specified SAX parser for stylesheet
  -?              Display this message
```

3. Make a backup copy of the standard iTPS XSL Templates

```
# cd /opt/ittm/Config/Templates/en/Payment
# cp subject.xsl subject_backup.xsl
# cp body.xsl body_backup.xsl
```

iTTM will read the files from the file system each time they are required.

4. Here is a standard acknowledgement XML file to prototype with

```
<Acknowledgement>
<NIB id="NIB_92BB680DAD6437C1FBAFF16B300236D92C465B52_1" version="2.0">
<ContextInfo msggrpid="71DD6AEF563B6D4867418705178509948524BBB1" msgid="BFI05">
</ContextInfo>
<StartTime>
<LocalTime id="LocalTime_92BB680DAD6437C1FBAFF16B300236D92C465B52_1"
time="20011105182855Z"/>
</StartTime>
<MsgTime>
<LocalTime id="LocalTime_92BB680DAD6437C1FBAFF16B300236D92C465B52_2"
time="20011105183621Z"/>
</MsgTime>
</NIB>
<Signature xmlns="http://www.w3.org/2000/02/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-hiroshi-dom-has
h-03.txt">
        </CanonicalizationMethod>
        <SignatureMethod Algorithm="http://www.w3.org/2000/02/xmldsig#rsa-sha1">
        </SignatureMethod>
    <Reference URI="#NIB_92BB680DAD6437C1FBAFF16B300236D92C465B52_1"><Transforms>
        <Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-hiroshi-dom-has
h-03.txt"></Transform>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/02/xmldsig#sha1">
        </DigestMethod>
        <DigestValue>1RBeHeKJkjlQq+/E6LogWfLF8QM=</DigestValue>
        </Reference>
        <Reference
URI="#ContentAcknowledgement_1576C34541070154DDE432BA7BF24D3B067BD99B_1">
        <Transforms>
            <Transform
Algorithm="http://search.ietf.org/internet-drafts/draft-ietf-trade-hiroshi-dom-has
h-03.txt"></Transform>
        </Transforms>
        <DigestMethod
Algorithm="http://www.w3.org/2000/02/xmldsig#sha1"></DigestMethod>
        <DigestValue>cnSVtgokSMyBaVPgd0HI/8LaDfU=</DigestValue>
    </Reference>
</SignedInfo>
<SignatureValue>DNkOnIoneexNzUqS9JhZV+ka1Yj7+fM8PtEFnkIcPDczg2ZNpFLgjlXthCAzbYf6q4
dRSYwXhYWbi/MJT2YC4XrhGtDTttEgcwyOz2mj1lCFsh77348RFvoeE351GWim9fHS3d4ozri9WJzpftVt
0hshVTiRBjvyZGex0l2zIU4=</SignatureValue>
    <KeyInfo>
```

```
        <X509Data>
            <X509IssuerSerial>
                <X509IssuerName>C=GB,O=iPlanet,OU=Payments Services,CN=Payments
Root</X509IssuerName>
                <X509SerialNumber>19</X509SerialNumber>
            </X509IssuerSerial>
        </X509Data>
    </KeyInfo>
    </Signature><ContentAcknowledgement
id="ContentAcknowledgement_1576C34541070154DDE432BA7BF24D3B067BD99B_1">
    <Header xml:lang="en">
        <Product>xPx</Product>
        <DocumentType>Acknowledgement</DocumentType>
        <Version>1.0</Version>
    </Header>
    <References>

<EleanorTransactionReference>1766631e9fdc51f1180002120448330</EleanorTransactionRe
ference>
</References>
    <AcknowledgementData>
    <AcknowledgementType>Service</AcknowledgementType>
        <Status>SUCCESS</Status>
        <ReasonCode>00ND00</ReasonCode>
        <ReasonText>Payment Instruction Details Accepted</ReasonText>
    </AcknowledgementData>
</ContentAcknowledgement>
</Acknowledgement>
```

5. Prototype your new XSL file using Instant Saxon or similar Java XSL toolkit. To translate an xml file to html issue the following (this assumes Instant Saxon is installed into Windows c:\temp)

   ```
   c:\temp\saxon acknowledgement.xml body.xsl > results.html
   ```

   This file can then be viewed in a standard web browser.

6. Once you are satisfied with the HTML output replace the iTTM stylesheets Copy your new XSL to the /opt/ittm/Config/Templates/en/Payment with your example, ensure your stylesheets are called body.xsl and subject.xsll

Biab BackEnd Messages

In order to design your own Biab Back End Messages, follow the same procedure as mentioned in the previous section using the following default template:

```
/opt/iws6/itps-biab-deploy/css/pretty.xsl
```

Glossary

Asynchronous Message (See also Eleanor Technical Specification and also Synchronous message) means that messages can be sent over a delayed time frame. Following a request, a response can be sent at any time. Whereas with a Synchronous message following a request the system waits until it gets a response.

Bank Name The name of the Scheme Member to whom this entry relates.

BIC-11 is an 11 digit Bank Identifier Code. The Bank Identifier Code is a unique address which, in telecommunication messages, identifies precisely the financial institutions involved in financial transactions. The BIC Directory Site can be found at http://www.bicdirectory.swift.com/.

BFI (See also Eleanor Technical Specification) The Buyers Bank or in Identrus terminology the IP.

BFIM (See also Eleanor Technical Specification) This is the model used for making a payment. The instruction is submitted by the submitting Corporate through the Buyer Financial Institution.

Biab Bank in a Box

Buyer The Buyer wants to purchase goods or services from the Seller. The Buyer is in possession of an Identrus smartcard. The Buyer is granted access to the payment facility by his bank. This is also abbreviated as BU in the Eleanor Scheme and as the SC in the Identrus four corner model

Buyers Bank The Buyer's Bank manages the Buyer's account from which payment is to be made. The Buyer's Bank issues the Buyer with his Identrus smartcard. The Buyer's Bank is a member of the Identrus schemes.

Cache A cache is a 'local' store used to hold recently accessed information, so that if further access is required a local copy may be used rather than a 'remote' copy. In most schemes this specifically refers to a local store containing copies of responses from the Identrus Root to certificate status checks.

Cancellation A cancellation is the revocation of a Payment Request. It may only be applied to a Payment Request, not the payment. Therefore payments that have been paid cannot be cancelled by this method.

Certified Payment Obligation (xPC) Where an Assured Payment is used, the Buyer requests the Buyer's Bank to underwrite (or assure) the payment to the Seller.

Certified Conditional Payment Obligation (CPC) This product has the same core characteristics as the Payment Obligation product, with the additional requirement that payment will not be released until evidence that all conditions attached to the payment have been met or

	waived. Since its certified the BFI is obliged to make the payment since the Buyer requests the Buyer's Bank to underwrite (or assure) the payment to the Seller.
Conditional Payment Order (CPx)	The Conditional Payment Order has the same core characteristics as the Payment Order product with the additional requirement that payment will not be released until evidence that all conditions attached to the payment have been met or waived. This product is also unilaterally revocable by the Buyer.
Conditional Payment Obligation (CPO)	The Conditional Payment Obligation has the same core characteristics as the payment Obligation product, with the additional requirement that payment will not be released until evidence that all conditions attached to the payment have been met or waived. As with the Payment Obligation, the Buyers Financial institution is not obliged to make the payment if, for instance, there are insufficient funds available in the Buyers account, and the product is bilaterally revocable.
CMS	iPlanet Certificate Management System
CPI	Corporate Payment Initiation API Library
Customer	For the purposes of this document this term describes a customer of a Member. of the Identrus Scheme.
FI Code	A BIC-11 code owned by the Scheme Member to whom this entry relates.
Identrus four-corner Model	This describes the four parties involved in an Identrus enabled transaction, namely two banks and their respective customers. Where two customers have the same bank, only three parties are involved, and this is known as a 'three party' model.
Identrus Root	The Identrus Root manages membership of the Identrus scheme. The Buyer's Bank and Seller's Bank verify each other's identity via the Identrus Root.
Issuing Participant (IP)	In the Identrus four-corner model this refers to the level 1 Identrus participant that issued the certificate to the Subscribing customer. This equates to "Buyer's Bank" in other payment transaction
iWS	iPlanet Web Server
iAS	iPlanet Application Server
iTTM	iPlanet Trustbase Transaction Manager
iMQ	iPlanet Message Queue
iMS	iPlanet Messaging System

iTPS iPlanet Trustbase Payment Services

JMS is an API. Many implementations of JMS are available from different providers (like IBM MQ and iMQ) and any of them can be used with iTPS

Membership The current status of the institution concerned. This can take the values Pending, Active, Suspended, Terminated.

Payment Order (xPx) A Payment order is a revocable, unconditional electronic instruction from the Buyer requesting the Buyer's Bank to execute a credit payment to the Seller on a specific date for a specified amount.

Payment Obligation (xPO) The commitment from a Buyer to pay a Seller a specified amount on a specified date. There is no obligation on the Buyers Financial Institution to pay if there are insufficient funds available on the due date. The instruction is only revocable bilaterally, i.e. with the assent of the current Holder of the obligation. This product is only supported in the SFI Model, although it will be possible to build it into a procurement application using the CPI library.

Relying Customer (RC) In the Identrus four-corner model this is the organisation that relies on the identity of the Subscribing customer in order to conduct trade. This equates to 'Seller' in a payment transaction.

Relying Participant (RP) This refers to the Relying Customer's bank. This equates to "Seller's Bank" in a payment transaction.

Service The current status of a service. This is used to mark temporary service interruptions. The allowed values are Available and Unavailable.

Seller The Seller wants to supply goods or services to the Buyer. The Seller is in possession of an Identrus private key. The Seller is granted access to the payment facility by his bank. This is also abbreviated as SE in the Eleanor Scheme and as the RC in the Identrus four corner model.

Sellers Bank The Seller's Bank manages the Seller's account into which payment is to be made. The Seller's Bank is a member of the Identrus schemes.

SFI (See also Eleanor Technical Specification) The Sellers Bank or in Identrus terminology the RP

SFIM (See also Eleanor Technical Specification) This is the model used for initiating a payment.The instruction is submitted by the submitting Corporate through the Seller Financial Institution. This is essentially the same as the Identrus four corner model.

Scheme Register	(See also Eleanor Technical Specification) The Eleanor Scheme has membership criteria. Each participating Financial Institution ensures that corresponding institutions are members of the scheme. It does so via an XML message that is referred to as a scheme register.
Subscribing Customer (SC)	In the Identrus four-corner model this is the end customer of the Identrus enabled service. This equates to the 'Buyer' in a payment transaction.
Synchronous Message	(See also Eleanor Technical Specification) See Asynchronous Message.(See also Eleanor Technical Specification and also Synchronous message) means that messages cannot be sent over a delayed time frame. Following a request, a response is sent immediately before the system can proceed. Whereas with an asynchronous message, following a request, a response can be sent at anytime.
Timeout	is the amount of time (in seconds) for which a client will wait for a server to respond before dropping the connection and returning with a timeout error. In the context of these screens, it is the amount of time the Seller's Bank will wait for the Buyer's Bank to respond.
Transaction	This refers to a complete end to end transaction, potentially involving multiple request-response pairs. Where a single message in such a transaction is referred to the term 'message' is used.
Transaction Reference	(See also Eleanor Technical Specification) The provision and handling of transaction references is key to allowing the Eleanor Payment products to be used effectively from third party systems and to assist in reconciliation of data

Index

F

G

H

I

J

M

O

P

R

S

T

U