



Payment Receipt Interface (PRI)

Revision 1.0.7

1. Introduction.....4

1.1 Registered Trademarks.....4

1.2 Who should read this document.....4

 1.2.1 Small merchants - a simple solution4

 1.2.2 Merchants who require full automatic payment verification.....4

1.3 Descriptions4

 1.3.1 Merchant4

 1.3.2 Customer4

 1.3.3 PRI4

1.4 About the PRI.....4

2. Basic description of the PRI process.....4

3. Payment Receipt Interface details.....5

3.1 Merchant account PRI settings.....5

3.2 HTML forms5

 3.2.1 Payment Request Form.....6

 3.2.2 Payment Status Notification Form10

 3.2.3 Payment Successful Form13

 3.2.4 Payment Cancelled Form.....14

3.3 Validation of critical payment information 15

 3.3.1 Using INPUT_HASH15

 3.3.2 Verify that no data tampering took place16

 3.3.2.1 Using PGP Signature Verification16

 3.3.2.2 Using PAYMENT_HASH Verification16

 3.3.3 Verify the payee account and amount received18

 3.3.4 Important security recommendations19

4. Payment Receipt Interface testing	19
Appendix A. Hash Algorithms.....	20
Appendix B. Payment Links	21
Appendix C. Change Log	22

1. Introduction

This document provides an overview of the Pecunix Payment Receipt Interface, and explains in detail how a merchant can easily accept payments in Pecunix gold from customers on an existing or new web site with this interface. To find out more about the Pecunix gold currency, see the web site at <http://pecunix.com>.

1.1 Registered Trademarks

The name '**Pecunix**' and '**money refined**' are registered trademarks of Pecunix Inc.

1.2 Who should read this document

1.2.1 Small merchants - a simple solution

If you are a small merchant who simply wishes to accept payment for individual items on a web site, it is not necessary to read this document. You can simply log on to your account, and in the account details use the Merchant Tools to "**create a link**" that you can simply copy and paste on to your web site. You will receive an e-mail message each time you receive a payment. For a description of how this link works, see [Appendix B](#) of this document.

1.2.2 Merchants who require full automatic payment verification

Any merchant who wishes to accept Pecunix payments from a web site and use scripting to verify payments will find this document useful and easy to understand. A small amount of basic technical knowledge would be helpful but is not necessary.

1.3 Descriptions

1.3.1 Merchant

In this document, anyone who wishes to receive Pecunix payments through a web site using this Payment Receipt Interface (PRI) is described as a merchant.

1.3.2 Customer

A customer is anyone who has a Pecunix account and wishes to pay a merchant through a merchant's web site using this Payment Receipt Interface (PRI)

1.3.3 PRI

The **Payment Receipt Interface** is abbreviated to **PRI**.

1.4 About the PRI

The PRI was designed to make it as easy and simple as possible for merchants to receive Pecunix payments through their web site. It uses existing well-known technology, and ensures industry standard security and encryption techniques to protect the data integrity and confidentiality of both the Merchant and the Customer. This gives the Customer a secure and "seamless" payment experience.

2. Basic description of the PRI process

This section provides a simple description of the PRI process from a customer's perspective.

- **Step 1:** The customer browses the merchant's web site
- **Step 2:** When the customer is ready to pay, (s)he proceeds to the merchant's checkout page

- **Step 3:** On the merchant's checkout page, the customer can choose to pay the merchant in Pecunix gold. (This page may present the total amount of the payment, which can be given in various currencies, including Pecunix gold.)
- **Step 4:** The customer clicks on the "pay with Pecunix" (or similar) button.
- **Step 5:** The customer's browser sends the information (and is directed) to the Pecunix secure payment web site.
- **Step 6:** The Pecunix system uses the information it receives from the customer's browser to generate a payment page that is presented to the customer.
- **Step 7:** The customer can enter the required payment details to complete the payment from his or her Account to the merchant Account, or (s)he can click Cancel to abort the payment process, in which case (s)he will be redirected back to a page on the merchant web site.
- **Step 8:** The customer enters the required payment details and completes the secure payment on the Pecunix web site.
- **Step 9:** After a successful payment is completed, the Pecunix system notifies the merchant system that the payment has been made, and also sends it some details regarding the payment. The payment details hash and digital signature (PGP) that is passed from the Pecunix system to the merchant system enables the merchant system to verify that the payment notification is authentic and came from the Pecunix system.
- **Step 10:** The customer is presented with a payment confirmation page on the Pecunix web site, showing all the details of the payment.
- **Step 11:** After the customer has viewed the payment result page, (s)he proceeds by clicking on Continue, and is directed back to the merchant's web site.

3. Payment Receipt Interface details

This section provides the details of how to accept payments with the PRI.

3.1 Merchant account PRI settings

For a merchant to accept Pecunix payments, the merchant must first have a Pecunix account into which they can accept payment. To open a Pecunix account please go to <https://secure.pecunix.com/money.refined...nac.newaccount1>

Merchants need to set up their Pecunix account to accept PRI payments as follows:

- Log in to the Pecunix account and select **Account Details>Merchant PRI**
- Enter a shared secret (8 to 50 characters) for use in the authentication hash (See 3.3 of this document).
- Select a hash algorithm for the payment authentication hash (SHA1 is the default).
- Select "who pays fees" (optional). By default the payee (merchant) will pay the spend fees. Use this setting to require the payer (customer) to pay the fees, or share the fees equally between the customer and merchant.
- Select "PRI payments only for this account" (optional). This option allows this account to only accept payments through the PRI and not accept payments made directly to the account through the web interface.
- Select "Require input hash" (optional). This option allows this account to only accept a PRI payment if a valid input hash is present.

3.2 HTML forms

The PRI utilises the following key HTML forms:

- **Payment Request Form** – This form is generated by the merchant system and submitted to the Pecunix system via the customer's browser.
- **Payment Status Notification Form** – This form is generated by the Pecunix system and submitted to the merchant system in a background process.

- **Payment Successful Form** -This form is generated by the Pecunix system and submitted to the merchant system via the customer's browser when (s)he clicks the continue button after a successful payment.
- **Payment Cancelled Form** - This form is generated by the Pecunix system and submitted to the merchant system via the customer's browser when (s)he clicks the cancel button before a payment is completed.

3.2.1 Payment Request Form

This form is generated by the merchant system and submitted to the Pecunix system via the customer's browser as described in **Step 4** of Section 2.

Action

The form action must be set to the following URL: <https://pri.pecunix.com/money.refined>

Method

The form method may be set to POST (recommended) or GET.

Fields

The following table shows the hidden fields on the **Payment Request Form**.

HTML Field Name and optional abbreviations	Field Description
PAYEE_ACCOUNT (required) or PAYE	The e-mail address of the merchant Pecunix account that will receive the payment.
PAYMENT_AMOUNT (required) or AMT	The amount of the payment in the designated payment units (see below). This amount must be greater than zero, and may be to an accuracy of 4 decimal places. Examples: 0.0041 or 5689.0215
PAYMENT_UNITS (optional) or UNIT	The payment units that the amount is specified in and to be used for the payment. (See Table 2 for a complete list of available payment units.) If this field is not present or is empty, the payment units will be set to grams (GAU).
WHO_PAYS_FEES (optional) or WPFEE	<p>Designates which account pays the payment fees. If this field is not present or is empty, the default is set and the recipient (merchant) pays the payment fees.</p> <p>The WHO_PAYS_FEES field value can be PAYER, PAYEE (default) or BOTH. The actions taken for each are as follows:</p> <p>PAYER – The customer pays the payment fees. The merchant does not pay any payment fees.</p> <p>PAYEE – The merchant pays the payment fees.</p> <p>BOTH – The payment fees are divided evenly between the customer and the merchant.</p>
STATUS_URL (optional) or SURL	<p>Designates whether Pecunix server should send payment status notification to the merchant server.</p> <p>If this field is not present or is empty, no status message will be sent to the merchant server. Note that in this case a payment notification e-mail generated by the Pecunix system will be sent if allowed in the merchant's account settings.</p> <p>Payment status is submitted as an HTML form POST upon successful completion of a Pecunix payment if a valid URL is specified as the value. The target URL would normally be that of a</p>

	<p>cgi script or other form processor. This URL can specify a secure protocol such as https. Example: https://gold-cart.com/paymentcatcher.php</p>
STATUS_TYPE (optional) or STYP	<p>Designates how the Pecunix server should send payment status to the merchant server. If the STATUS_URL field is not present or is empty this field is ignored. If this field is not present, the default value of FORM is used.</p> <p>The STATUS_TYPE field value can be FORM (default) or XML. The actions taken for each are as follows:</p> <p>FORM – The payment status is sent to the STATUS_URL in an HTML form using the POST method.</p> <p>XML - The payment status is sent to the STATUS_URL as an XML packet, signed by PGP and submitted in the BODY of the message.</p>
PAYMENT_URL (optional) or PURL	<p>The URL to which the customer is sent (and a form is optionally submitted) after a successful Pecunix payment to the merchant.</p> <p>If this field is not present or is empty, the Pecunix system will attempt to set this value to the address of the referring page.</p> <p>This is the customer's normal return path to the merchant's web site. This URL can specify a secure protocol such as https. By default, this URL is assumed to be a simple LINK and no data is submitted. Other actions are possible when the optional PAYMENT_URL_METHOD field is specified (see below).</p>
PAYMENT_URL_METHOD (optional) or PUM	<p>If no PAYMENT_URL field is specified, this field is ignored. When a value is set for the PAYMENT_URL, the PAYMENT_URL_METHOD value can be set to POST GET or LINK. The actions taken for each are as follows:</p> <p>POST – The customer is returned to the PAYMENT_URL and payment information is sent in an HTML form using the POST method.</p> <p>GET - The customer is returned to the PAYMENT_URL and payment information is sent in an HTML form using the GET method.</p> <p>LINK - The customer is returned to the PAYMENT_URL and no payment information is sent.</p>
NOPAYMENT_URL (optional) or NPURL	<p>The URL to which the customer is sent (and a form is optionally submitted) after an unsuccessful or cancelled Pecunix payment to the merchant.</p> <p>If this field is not present or is empty, the Pecunix system will attempt to set this value to the address of the referring page.</p> <p>This is the customer's alternate return path to the merchant's web site. This URL can specify a secure protocol such as https. By default, this URL is assumed to be a simple LINK and no data submitted. Other actions are possible when the optional NOPAYMENT_URL_METHOD field is specified (see below).</p>
NOPAYMENT_URL_METHOD (optional) or NPUM	<p>If no NOPAYMENT_URL field is specified, this field is ignored. When a value is set for the NOPAYMENT_URL, the NOPAYMENT_URL_METHOD value can be set to POST GET or</p>

	<p>LINK. The actions taken for each are as follows:</p> <p>POST - The payment status is sent to the NOPAYMENT_URL in an HTML form using the POST method.</p> <p>GET - The payment status is sent to the NOPAYMENT_URL in an HTML form using the GET method.</p> <p>LINK - The customer is returned to the NOPAYMENT_URL and no payment information is sent.</p>
INPUT_HASH (optional) or HASH	<p>The value of this field is a message digest (MD5 or SHA1) calculated from the values of the PAYEE_ACCOUNT, PAYMENT_AMOUNT, PAYMENT_UNITS, PAYMENT_ID, WHO_PAYS_FEES and your shared secret. (See section 3.3 below.)</p> <p>If this field is present, the Pecunix system will check the input of the payment request form to ensure that no tampering or changes have been made to the PAYEE_ACCOUNT, PAYMENT_UNITS, PAYMENT_ID, WHO_PAYS_FEES or PAYMENT_AMOUNT values.</p>
PAYMENT_ID (optional) or PID	<p>The value of this field attaches an order number or any other reference number to the payment. This field is included when calculating the authentication hash sent to the status URL. If the field is not present, an empty string is used as its value when computing the authentication hash. This field accepts up to 10 digits only from 0 to 9.</p>
SUGGESTED_MEMO (optional) or SMEM	<p>If this input field is present, the memo area of the payment form is pre-filled in with its value. At most, 100 characters can be entered into the memo field. The customer cannot edit the memo but can add a message at the time of the payment.</p>
Extra merchant parameters (optional)	<p>Up to five optional text fields, each with a maximum length of 150 characters, may be included in the form. These text fields can have any name other than the names already listed above and will be carried through the Pecunix system, to eventually be returned to the merchant system via the normal return form or the alternate return form. Note: If more than 5 fields are included, the excess fields will be discarded.</p>

Table 1. Hidden Fields On The Payment Request Form.

Payment Units

The following table lists the valid currency codes that the PRI will accept on the **Payment Request Form**.

Payment Unit	PAYMENT_UNITS Value
Grams (default Value)	GAU
Troy Ounces	OAU
The Pecunix PRI also supports all payment units listed at http://pecunix.com/money.refined...gld.rates	

Table 2: Payment Units

If the specified payment unit is not GAU or OAU (a weight of gold), the Pecunix system will convert the specified amount into grams of Pecunix on behalf of the merchant using Pecunix's exchange rates. If the merchant submits a payment unit that requires conversion, the merchant accepts that the exchange rate supplied by Pecunix will be satisfactory. Pecunix uses a conversion factor of 31.034768 grams per troy ounce in cases where the unit specified is OAU.

Example 1

The following is an example of how a **Payment Request Form** might look.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Sample Pecunix PRI Payment Request Form</title>
</head>

<body>

<form action="https://pri.pecunix.com/money.refined" method="post">

<!-- Required Fields Start -->
  <input type="hidden" name="PAYEE_ACCOUNT" value="charityfund@pecunix.com">
  <input type="hidden" name="PAYMENT_AMOUNT" value="1.00">
<!-- Required Fields End -->

<!-- Optional Fields Start -->
  <input type="hidden" name="PAYMENT_URL" value="http://pecunix.com/money.refined...ref.developer">
  <input type="hidden" name="NOPAYMENT_URL" value="http://pecunix.com/money.refined...ref.developer">
  <input type="hidden" name="STATUS_URL" value="https://gold-cart.com/paymentcatcher.php">
  <input type="hidden" name="STATUS_TYPE" value="FORM">

  <input type="hidden" name="PAYMENT_URL_METHOD" value="POST">
  <input type="hidden" name="NOPAYMENT_URL_METHOD" value="POST">

  <input type="hidden" name="INPUT_HASH" value="A1213419A0C503FC2767891F21463751">
  <input type="hidden" name="PAYMENT_UNITS" value="AUD">
  <input type="hidden" name="WHO_PAYS_FEES" value="PAYER">
  <input type="hidden" name="PAYMENT_ID" value="1234">
  <input type="hidden" name="SUGGESTED_MEMO" value="Payment to gold-cart.com">
<!-- Optional Fields End -->

  <input type="submit" value="Pay with Pecunix">
</form>

</body>
</html>

```

3.2.2 Payment Status Notification Form

This form is generated by the Pecunix system and submitted to the merchant's system in a background process as described in **Step 9** of Section 2. It is VERY IMPORTANT that the merchant system verifies the information submitted by this form using methods described in Section 3.3 below.

Action

The form action is set to the STATUS_URL given in the **Payment Request Form**.

Method

The form method is POST.

No response is expected from the merchant system when this form is submitted and if any HTML is returned it is ignored and discarded. The Pecunix system will resubmit the **Payment Status Notification Form** three times or until it obtains a valid http(s) status response indicating a successful post. Because of this, under certain communication error conditions it may be possible for the merchant's system to receive multiple POSTs of the **Payment Status Notification Form**.

Payment Status Information

The following table shows the information returned by the **Payment Status Notification Form**.

Information	Description
PAYEE_ACCOUNT	Same value sent by merchant system. (The e-mail address of the Pecunix account that received the payment.)
PAYMENT_AMOUNT	Same value sent by merchant system. This amount is always numeric and to an accuracy of 4 decimal places.
PAYMENT_UNITS	Same value sent by merchant system.
PAYMENT_REC_ID	The transaction number allocated to this payment by the Pecunix system.
PAYER_ACCOUNT	The customer's e-mail address.
PAYMENT_HASH	Either an MD5 or SHA1 hash calculated from the information in this form. This hash is always in upper case. (See section 3.3 below.)
PAYMENT_GRAMS	The actual amount of the payment in grams and formatted to 4 decimal places.
PAYMENT_ID	Same value sent by merchant system, or an empty string if not sent by merchant.
PAYMENT_FEE	The actual fee charged to the merchant account for this payment. This amount is always numeric and to an accuracy of 6 decimal places.
SUGGESTED_MEMO (optional) or SMEM	This is any text sent as SUGGESTED_MEMO, plus any message the customer may have entered at the time of the payment.
TXN_DATETIME	The date & time that the payment was made, in the ISO 8601 format "1995-02-04 22:45:00" (UTC). (YYYY-MM-DD HH:mm:ss)
Extra merchant parameters	Up to five optional text fields, each with a maximum length of 150 characters with the same names and values as sent by the merchant system on the Payment Request Form.

Table 3. Payment Status Notification Form Information

Status Information Type

The Information sent to STATUS_URL is presented in one of 3 ways depending on the STATUS_TYPE set in the **Payment Request Form**.

Example 2

The following is an example of a **Payment Status Notification Form** that is sent to the merchant system by the Pecunix system when the STATUS_TYPE is set to FORM.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Sample Pecunix PRI Payment Status Notification Form</title>
</head>

<body>

<form action="<STATUS_URL> as given in the payment request form" method="post">

  <input type="hidden" name="PAYEE_ACCOUNT" value="charityfund@pecunix.com">
  <input type="hidden" name="PAYMENT_AMOUNT " value="1.00">
  <input type="hidden" name="PAYMENT_UNITS" value="AUD">
  <input type="hidden" name="PAYMENT_REC_ID" value="000014568">
  <input type="hidden" name="PAYER_ACCOUNT" value="customer@gold-cart.com">
  <input type="hidden" name="PAYMENT_HASH" value="LKJH423HJTLWKJ23L4K5J234LKJ9U54F">
  <input type="hidden" name="PAYMENT_GRAMS" value="0.0540">
  <input type="hidden" name="PAYMENT_ID" value="1234">
  <input type="hidden" name="PAYMENT_FEE" value="0.0002">
  <input type="hidden" name="TXN_DATETIME" value="2002-04-10 10:14:54">
  <input type="hidden" name="SUGGESTED_MEMO" value="Payment to gold-cart.com">
  <input type="hidden" name="Extra merchant parameters" value="as given in payment request form">

</form>

</body>
</html>
```

Example 3

The following is an example of the information that is sent to the merchant system by the Pecunix system when the STATUS_TYPE is set to XML.

The information will be presented in the HTTP message body. This information will be signed by the Pecunix private PGP key. You can get the Pecunix public PGP key to use for verification at <http://pecunix.com/money.refined...ref.pgpkey>

NOTE:

This is also an example the e-mail that is sent to the merchant if the merchant has not included a STATUS_URL in the Payment **Status Notification Form** and the settings in the merchant's Pecunix account allow e-mail notification.

```
<TransferResponse>
<Signature type="PGP">
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

</Signature>
<Receipt>
  <ReceiptId> 000014568 </ReceiptId>
  <Date> 2002-04-10 10:14:54 </Date>
  <PayerName> gold-cart </PayerName>
  <Amount> 0.0540 </Amount>
  <Fee>
    <TransferFee> 0.0002 </TransferFee>
  </Fee>
  <Transfer>
    <TransferId> </TransferId>
    <Payer> customer@gold-cart.com </Payer>
    <Payee> charityfund@pecunix.com </Payee>
    <CurrencyId> GAU </CurrencyId>
    <Equivalent>
      <CurrencyId> AUD </CurrencyId>
      <Amount> 100.00 </Amount>
    </Equivalent>
    <FeePaidBy> Payee </FeePaidBy>
    <Memo> Your message goes here </Memo>
  </Transfer>
</receipt>
<PaymentHash> LKJH423HJTLWKJ23L4K5J234LKJ9U54F </PaymentHash>
<MerchantParameters>
  <Parameter name=""></Parameter>
  <Parameter name=""></Parameter>
</MerchantParameters>
<Signature type="PGP">
-----BEGIN PGP SIGNATURE-----
Version: PECUNIX "Money Refined" - <http://pecunix.com>

iQA/AwUBO2dEYFd1EPkDbqR2EQJBNACggzRpGad9fL/QymztGT6amHK1kqEAoLdV
VF1xXqpoXg2e5/kWA3ROpRji
=e2/J
-----END PGP SIGNATURE-----
</Signature>
</TransferResponse>
```

NOTE: The signature on this example does not verify correctly due to formatting adjustments made for readability.

3.2.3 Payment Successful Form

This form is generated by the Pecunix system and is submitted to the merchant system (via the customer's browser) as described in **Step 11** of Section 2, after the successful completion of a payment.

Action

The form action is set to the PAYMENT_URL given in the **Payment Request Form**.

Method

The form method is set to LINK (default), POST or GET depending on the value specified in PAYMENT_URL_METHOD of the **Payment Request Form**. In the case of a LINK, no form data will be sent to the PAYMENT_URL on the merchant system, but will link to it (via the Continue button).

Fields

The following table describes the hidden fields on the **Payment Successful Form**.

HTML Field Name	Field Description
PAYEE_ACCOUNT	Same value sent by merchant system. (The e-mail address of the Pecunix account that received the payment.)
PAYMENT_AMOUNT	Same value sent by merchant system. This amount is always numeric and to an accuracy of 4 decimal places.
PAYMENT_UNITS	Same value sent by merchant system.
PAYMENT_REC_ID	The transaction number allocated to this payment by the Pecunix system.
PAYER_ACCOUNT	The customer's e-mail address.
PAYMENT_ID	Same value sent by merchant system, or an empty string if not sent by merchant.
ACTUAL_PAYMENT_GRAMS	The actual grams of Pecunix transferred for this payment. This value will be the same as PAYMENT_AMOUNT if the unit for the payment was GAU.
Extra merchant parameters	Up to five optional text fields, each with a maximum length of 150 characters with the same names and values as sent by the merchant system on the Payment Request Form.

Table 4. Hidden Fields On The Payment Successful Form

Example 4

The following is an example of a **Payment Successful Form** that is sent to the merchant system by the Pecunix system when the customer clicks "continue" after a payment has been completed successfully.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Sample Pecunix PRI Payment Successful Form </title>
</head>

<body>

<form action="<PAYMENT_URL> as given in the payment request form" method="<PAYMENT_URL_METHOD">

  <input type="hidden" name="PAYEE_ACCOUNT" value="charityfund@pecunix.com">
  <input type="hidden" name="PAYMENT_AMOUNT" value="1.00">
  <input type="hidden" name="PAYMENT_UNITS" value="AUD">
  <input type="hidden" name="PAYMENT_REC_ID" value="000014568">
  <input type="hidden" name="PAYER_ACCOUNT" value="customer@gold-cart.com">
  <input type="hidden" name="PAYMENT_ID" value="1234">
  <input type="hidden" name="ACTUAL_PAYMENT_GRAMS" value="0.5314">
  <input type="hidden" name="Extra merchant parameters" value="as given in payment request form">

</form>

</body>
</html>
```

3.2.4 Payment Cancelled Form

This form is generated by the Pecunix system and is submitted to the merchant system (via the customer's browser) as described in **Step 7** of Section 2, after the customer cancels the payment.

Action

The form action is set to the NOPAYMENT_URL given in the **Payment Request Form**.

Method

The form method is set to LINK (default), POST or GET depending on the value specified in NOPAYMENT_URL_METHOD of the **Payment Request Form**. In the case of a LINK, no form data will be sent to the NOPAYMENT_URL on the merchant system, but will link to it (via the Continue button).

Fields

The following table describes the hidden fields on the **Payment Cancelled Form**.

HTML Field Name	Field Description
PAYEE_ACCOUNT	Same value sent by merchant system. (The e-mail address of the Pecunix account that would have received the payment.)
PAYMENT_AMOUNT	Same value sent by merchant system. This amount is always numeric and to an accuracy of 4 decimal places.
PAYMENT_UNITS	Same value sent by merchant system.
PAYMENT_REC_ID	This value is set to 0 (zero).
PAYMENT_ID	Same value sent by merchant system, or an empty string if not sent by merchant.
Extra merchant parameters	Up to five optional text fields, each with a maximum length of 150 characters with the same names and values as sent by the merchant system on the Payment Request Form.

Table 5. Hidden Fields On The Payment Cancelled Form

Example 5

The following is an example of a **Payment Cancelled Form** that is sent to the merchant system by the Pecunix system when the customer clicks "cancel" during a payment.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Sample Pecunix PRI Payment Cancelled Form </title>
</head>

<body>

<form action="<NOPAYMENT_URL> as given in the payment request form" method="<NOPAYMENT_URL_METHOD">

  <input type="hidden" name="PAYEE_ACCOUNT" value="charityfund@pecunix.com">
  <input type="hidden" name="PAYMENT_AMOUNT" value="1.00">
  <input type="hidden" name="PAYMENT_UNITS" value="AUD">
  <input type="hidden" name="PAYMENT_REC_ID" value="0">
  <input type="hidden" name="PAYMENT_ID" value="1234">
  <input type="hidden" name="Extra merchant parameters" value="as given in payment request form">

</form>

</body>
</html>

```

3.3 Validation of critical payment information

The PRI utilises cryptographic validation in the following key HTML forms:

- **Payment Request Form** – This form is generated by the merchant system and submitted to the Pecunix system via the customer's browser. Although it is not required, it is recommended that the merchant system include an INPUT_HASH field in this form to prevent any possible tampering with the values of the PAYEE_ACCOUNT, PAYMENT_AMOUNT, WHO_PAYS_FEES and PAYMENT_UNITS fields before the form is submitted.
- **Payment Status Notification Form** – This form is generated by the Pecunix system and submitted to the STATUS_URL on the merchant system in a background process after the successful completion of a payment from a customer (See 3.2.2 above). Although not required, it is recommended that the merchant system performs the following validations:

1. Verify that no data tampering took place.
2. Verify the payee account and amount received (not necessary if INPUT_HASH was used).

3.3.1 Using INPUT_HASH

Due to the nature of HTML, it is possible for a dishonest customer to adjust the values of the form fields in the payment request form before it is submitted to the Pecunix system for processing. To ensure this is not possible on the payment critical fields and to make verification of the payment easier, it is recommended that the merchant use the INPUT_HASH. The input hash may be an MD5 hash or an SHA1 hash.

Constructing the INPUT_HASH

Construct the input hash by simply combining (concatenating) the relevant data in a string with colons as delimiters and then performing the hash calculation on the string. There is a simple tool in your Pecunix account under Account Details > Merchant PRI that will help you to create the INPUT_HASH.

Create the string as follows:

PAYEE_ACCOUNT:PAYMENT_AMOUNT:PAYMENT_UNITS:PAYMENT_ID:WHO_PAYS_FEES:Shared secret

Note that the PAYEE_ACCOUNT e-mail address must be lower case, the PAYMENT_UNITS and WHO_PAYS_FEES must be upper case, and that the PAYMENT_ID and shared secret are case sensitive.

Example 6

The following are examples of how to construct the INPUT_HASH

```
If PAYMENT_ID and WHO_PAYS_FEES are defined:
charityfund@pecunix.com:1.00:AUD:1234:PAYER:My shared secret
The resulting MD5 hash is: D6FCB7D3AD15CB97C4C094D9FC769BEF
The resulting SHA-1 hash is: 72D5AADDE2A2172693B5185B8B66463508C9B632
-----
If PAYMENT_ID and WHO_PAYS_FEES are not defined:
charityfund@pecunix.com:1.00:AUD:::My shared secret
The resulting MD5 hash is: A7A55656E745ED25C5115D7C4D9C60AD
The resulting SHA-1 hash is: 9581D5683F017314F85BA9D96461C62697CF3A40
```

NOTE: As seen in the second example, if you do not wish to include an optional PAYMENT_ID in your form input, you must still include the delimiters for the space where it would be when creating the INPUT_HASH.

3.3.2 Verify that no data tampering took place

3.3.2.1 Using PGP Signature Verification

When the Pecunix system sends the payment notification details to the merchant system as a WDDX packet or in an e-mail, the information is signed using the Pecunix private key. If the PGP signature verifies correctly using the Pecunix public key the merchant can be sure of the integrity of the details in the message and that they do originate from the Pecunix system. The Pecunix public key is available at <http://pecunix.com/money.refined...ref.pgpkey>

IMPORTANT: If the merchant did not use the optional INPUT_HASH in the payment request form, it is still necessary to check that the PAYEE_ACCOUNT, the PAYMENT_AMOUNT and the PAYMENT_UNITS are correct.

To find out more about using PGP please visit <http://pecunix.com/money.refined...edu.welcome>

3.3.2.2 Using PAYMENT_HASH Verification

When the Pecunix system sends the payment notification details to the merchant system as a WDDX packet or as a form POST, one of the details is a hash (message digest) of the payment notification details. The merchant can validate the values of the payment notification details by reconstructing the hash using the values given and comparing the 2 hashes. If the hashes are identical, the merchant can be sure of the integrity of the details in the message and that they do originate from the Pecunix system.

IMPORTANT: If the merchant did not use the optional INPUT_HASH in the payment request form, it is still necessary to check that the PAYEE_ACCOUNT, the PAYMENT_AMOUNT and the PAYMENT_UNITS are correct.

PAYMENT_HASH Details

The Pecunix system generates the PAYMENT_HASH by concatenating the following fields in the **Payment Status Notification Form** using colons as delimiters:

1. PAYEE_ACCOUNT (lower case)
2. PAYMENT_AMOUNT (always to 2 decimal places)
3. PAYMENT_UNITS (upper case)
4. PAYER_ACCOUNT (lower case)
5. PAYMENT_REC_ID
6. PAYMENT_GRAMS (always to 4 decimal places)
7. PAYMENT_ID (case sensitive)(if no PAYMENT_ID was sent, use an empty string)
8. PAYMENT_FEE (always to 4 decimal places)
9. TXN_DATETIME (ISO 8601 format)
10. Shared secret (from your account settings) (case sensitive)

The Pecunix system generates the type of hash selected in the merchant's account settings (See section 3.1). An MD5 hash is 32 upper-case hexadecimal digits long and an SHA-1 hash (recommended) is 40 upper-case hexadecimal digits long.

Example 8

The following are examples of how to construct the PAYMENT_HASH

Details of the payment:

```
PAYEE_ACCOUNT - charityfund@pecunix.com
PAYMENT_AMOUNT - 1.00
PAYMENT_UNITS - AUD
PAYER_ACCOUNT - customer@gold-cart.com
PAYMENT_REC_ID - 000014568
PAYMENT_GRAMS - 0.0540
PAYMENT_ID - 1234
PAYMENT_FEE - 0.0002
TXN_DATETIME - 2002-04-10 10:14:54
```

```
charityfund@pecunix.com:1.00:AUD:customer@gold-cart.com:000014568:0.0540:1234:18.4913:0.0002:2002-04-10 10:14:54:My shared secret
```

The resulting MD5 hash is: 805DECB97439A947C1B9B62D2FBD3C7D

The resulting SHA-1 hash is: 2B345487968795E809C3F6478549C81F31FE94B6

Details of a payment where no PAYMENT_ID is given:

```
PAYEE_ACCOUNT - charityfund@pecunix.com
PAYMENT_AMOUNT - 1.00
PAYMENT_UNITS - AUD
PAYER_ACCOUNT - customer@gold-cart.com
PAYMENT_REC_ID - 000014568
PAYMENT_GRAMS - 0.0540
PAYMENT_ID -
PAYMENT_FEE - 0.0002
TXN_DATETIME - 2002-04-10 10:14:54
```

```
charityfund@pecunix.com:1.00:AUD:customer@gold-cart.com:000014568:0.05409::18.4913:0.0002:2002-04-10 10:14:54:My shared secret
```

The resulting MD5 hash is: CAD3A23D2C53609CA7636B38FD8C56EF

The resulting SHA-1 hash is: 70BB2881C726D501FDF6D1DD5DF6AABA8656C5C7

NOTE: The combined (concatenated) line of details in these examples have been wrapped to a second line because of space constraints. In practise, there are no line breaks in the string.

3.3.3 Verify the payee account and amount received

Due to the nature of HTML, it is possible for a dishonest customer to adjust the values of the form fields (such as the amount, payment units and payee account) in the payment request form before it is submitted to the Pecunix system for processing. To ensure this is not possible on the payment critical fields and to make verification of the payment easier, it is recommended that the merchant use the INPUT_HASH. If the merchant chooses not to use the INPUT_HASH it is very important that the merchant system verifies that the PAYEE_ACCOUNT, PAYMENT_AMOUNT and PAYMENT_UNITS are correct and as expected, even if the PAYMENT_HASH verifies correctly.

3.3.4 Important security recommendations

Pecunix strongly recommends that the merchant should use the INPUT_HASH to ensure the data integrity of incoming payment requests. It is also recommended that a secure (SSL) URL is used for the STATUS_URL to help maintain data security and that SHA-1 hash is used rather than MD5 hash.

4. Payment Receipt Interface testing

The best way to test your PRI implementation is to use very small payment amounts (e.g. 0.0001 GAU). This allows you to test using the live Pecunix site in a real-world situation. Before live deployment, the PAYMENT_AMOUNT and INPUT_HASH can then be adjusted to the real values required in your entry form.

You are encouraged to join the developers' e-mail discussion list and ask questions or make comments. To join send an e-mail to lists@pecunix.com with "join developers@pecunix.com" in the subject line.

Appendix A. Hash Algorithms

A one-way hash function, also known as a message digest, fingerprint or compression function, is a mathematical function which takes a variable-length input string and converts it into a fixed-length binary sequence. Furthermore, a one-way hash function is designed in such a way that it is hard to reverse the process, that is, to find a string that hashes to a given value (hence the name one-way). A good hash function also makes it hard to find two strings that would produce the same hash value.

Hash algorithms are generally used to create digital signatures that can be used to verify that data has not been changed since the signature was created. Merchants and developers using the Pecunix system can choose whether they wish to implement either an MD5 or SHA-1 hash.

Ron Rivest invented the MD5 hash and the reference implementation is published as Internet RFC 1321. MD stands for Message Digest and the algorithm produces a 128-bit hash value.

SHA-1 stands for Secure Hash Algorithm 1 and NIST and the NSA designed it. SHA-1 produces 160-bit hash values, longer than MD5, and SHA-1 is generally considered more secure than other algorithms. SHA-1 is the recommended hash algorithm and its definition can be found at <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.

Appendix B. Payment Links

A simple link placed on your website or in an e-mail message, allows your customers to click through to the Pecunix website to pay for your products. A link has all the information that is required to make the payment and makes it simple for anyone to accept Pecunix payments for their goods or services. This feature has excellent applications, for example bulk billing by e-mail.

A link takes the customer to a secure payment page on the Pecunix web site with all the correct details of the transaction already confirmed by the system. The customer simply completes the login process and the payment is completed.

Creating a payment link

To create a payment link, list the payment information in the correct order with commas between each value as follows:

1. Payee e-mail address.
2. Amount to pay.
3. Three letter currency code (optional, default is GAU, see [Table 2](#)).
4. Who pays the fees (optional, default is PAYEE, possible values are PAYER, PAYEE or BOTH).
5. A reference number (optional).
6. Input hash (optional, see [section 3.3.1](#) above).

Example 9

The following are examples of **Payment Links**.

```
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1.00,AUD
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1.00,AUD,PAYER
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1.00,AUD,BOTH,1234
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1.00,AUD,,1234
https://pri.pecunix.com/money.refined?payment=fun@pecunix.com,1.00,,,1234
https://pri... ..refined?payment=fun@pecunix.com,1.00,,,, 82F3F622AAB0817D5151E722FDA4DA98
```

Appendix C. Change Log

2002-11-07

Amended Table 2: Payment Units to include all rates listed at <http://pecunix.com/money.refined...gld.rates>

2002-09-11

Added <Signature> parameter in the xml response.

2002-09-15

Added Appendix B – Payment Links.

2003-05-13

Corrected the hash values in Example 6, using INPUT_HASH.