

# Callback guide

Version 2.0.1

### **Copyright**

© SecureTrading 2006. All rights reserved. No part of this document may be photocopied, reproduced, stored in a retrieval system or transmitted in any form or by any means whether electronic, mechanical or otherwise without the prior written permission of SecureTrading Ltd.

### **Disclaimer**

This document is for informational purposes only. SecureTrading make no warranties, express or implied, through the distribution of this document. No warranty of accuracy is given concerning the contents of the information contained in this publication. To the extent permitted by law no liability (including liability to any person by reason of negligence) will be accepted by SecureTrading, its subsidiaries or employees for any direct or indirect loss or damage caused by omissions from or inaccuracies in this document.

SecureTrading reserves the right to revise the content without obligation to notify any person of such changes.

Document revised on 03-Aug-2006.

## Contents

1	Introduction.....	4
2	The transaction process using callback.....	5
3	Callback configuration.....	6
3.1	callback.txt and callback-f.txt.....	6
3.1.1	method1.....	6
3.1.2	url1.....	6
3.1.3	fields1.....	6
3.2	Multiple scripts.....	7
3.3	Callbackurl.....	7
3.4	failureurl.....	8
4	Example of callback.....	9
4.1	Installing the script.....	9
4.2	callback.txt setup.....	9
4.3	callbackurl setup.....	10
4.4	Testing.....	10
4.5	Logging failed authorisations.....	11
5	Further customisation.....	12
5.1	pipe1, pipe2, etc.....	12
5.2	email1, email2 etc.....	12
6	Using callback.....	14
7	Troubleshooting.....	15

---

## 1 Introduction

The **ST Payment Pages callback** feature allows you to run a script on your merchant web server at the final stage of a transaction. Different scripts can be run in different circumstances, for example whether or not the transaction is authorised by the bank.

**Callback** is an advanced feature of SecureTrading's **ST Payment Pages** service, and we recommend that you only use it if you are confident at writing scripts, and you can execute these scripts on your merchant web server.

**Callback** can post data back to most programming languages.

## 2 The transaction process using callback

The initial stages of the transaction process are the same whether or not you are using **callback**:

1. The customer submits the order form from the merchant web server. This form calls `form.cgi` on the SecureTrading secure server, which sends the appropriate `form.html` to the customer.
2. The customer enters payment details on `form.html` and submits the form. This form calls `process.cgi` on the SecureTrading secure server, which controls the next steps of the transaction process.
3. The SecureTrading secure server sends the payment details to a SecureTrading payment gateway, which in turn sends the details to the appropriate financial institution for authorisation. The results of the authorisation request are passed back from the bank, via the SecureTrading gateway, to the SecureTrading secure server.

Depending on the results of the authorisation request, one of two chains of events will take place. Either:

- 4a. If the authorisation is successful, the SecureTrading secure server sends `success.html` to the customer, and also emails `customeremail.txt` and `merchantemail.txt` to the customer and to the merchant respectively.

Or:

- 4b. If the authorisation is unsuccessful, SecureTrading secure server sends `failure.html` to the customer, and emails `failureemail.txt` to the merchant.

**Callback** adds extra functionality to steps 4a and 4b. In addition to (or instead of) sending web pages and emails to the customer and to the merchant, the SecureTrading secure server can call (and pass data to) one of up to eight scripts located on the merchant web server.

The scripts on the merchant server can use most of the data passed from `form.html`, and also the data generated by SecureTrading and the bank during the authorisation process. The only data you do not have access to is credit/debit card details. **Callback** uses the same fields that are used in the other steps of the transaction process – for a full list of these, see the reference section of the **ST Payment Pages Setup Guide**.

For example, suppose you have a script on your server to update your database of customer records. The SecureTrading secure server can call your script whenever your customer makes a transaction, passing to it information such as customer name, address and phone numbers. Your script can now update your database. Other things you might want your script to do are update your stock records, produce a summary printout, or, as in the example given in this guide, produce a summary log file of the transaction.

You can run scripts following successful authorisations only, following unsuccessful authorisations only, or following all authorisations, as you require. For instance, if you only want to run a script following a successful authorisation, then the SecureTrading secure server will run the appropriate script on your merchant server after a successful authorisation, and will carry out its default sequence of events (i.e. show `failure.html` and send `failure.txt`) after an unsuccessful authorisation.

## 3 Callback configuration

The operation of **callback** is controlled by:

- Two files, `callback.txt` and `callback-f.txt`, which are stored on the SecureTrading secure server. Examples of these files can be downloaded from the SecureTrading website at <http://www.securetrading.com/index.cfm?alias=dl-cback>
- Two variables, `callbackurl` and `failureurl`, which are passed through `form.html` to `process.cgi`.

### 3.1 `callback.txt` and `callback-f.txt`

`callback.txt` contains information about which script to run on your merchant server if an authorisation is successful. `callback-f.txt` contains similar information for unsuccessful authorisations. After these files have been created, they should be uploaded (using My-ST) to the SecureTrading secure server. The two files have the same format, so we'll only describe `callback.txt` here. You can create a `callback-f.txt` file in exactly the same way as `callback.txt`.

A basic `callback.txt` file looks like this:

```
method1  POST
url1     http://www.merchantwebserver.com/merchant/cgi-bin/successscript.cgi
fields1  orderref, name, address, postcode, formattedamount, stauthcode,
         securitymessage
```

The format of each line is:

*variable<tab>data*

(where <tab> is a tab character). The variable at the start of each line (`method1`, `url1`, `fields1`) **must** be separated from the data by a **single tab character**, not by spaces. In addition, each variable/data pair **must** be on a separate line. **Callback** will not work correctly, otherwise.

The variables specify the method used to pass data to the script on the merchant web server, the URL of the script that is to be run on the merchant web server and the data that is to be passed to the script. All three of these variables must be specified before **Callback** will run the script.

#### 3.1.1 `method1`

This variable defines the method by which data is passed to the script on the merchant server. The methods available are the same as are used in HTML forms; POST being the method usually used.

#### 3.1.2 `url1`

This variable contains the URL of the script to be run on the merchant web server.

#### 3.1.3 `fields1`

This variable contains a list of the fields that should be sent to the script on the merchant web server. Individual field names are separated by a comma followed by a single space. You can specify any or all (except for credit card details) of the variables that are passed from `form.html` to `process.cgi` during step 2 of the transaction process. The fields used in the examples in the document are a snapshot of the information accessible to the merchant. For the full list of available fields please refer to the ST Payment Pages document.

## 3.2 Multiple scripts

You can run one of up to eight scripts on your merchant web server following an authorisation (the actual script that is run is controlled by the variables `callbackurl` and `failureurl` – see later). To include the necessary information about further scripts in `callback.txt` and `callback-f.txt`, use the variables `method2`, `url2`, `fields2`; `method3`, `url3`, `fields3`; up to `method8`, `url8`, `fields8`; like this:

```
method1 POST
url1      http://www.merchantwebserver.com/merchant/cgi-bin/script_a.cgi
fields1  orderref, name, securitymessage, formattedamount, stauthcode
method2 POST
url2      http://www.merchantwebserver.com/merchant/cgi-bin/script_b.cgi
fields2  orderref, name, address, postcode, formattedamount, stauthcode, color
method3 POST
url3      http://www.merchantwebserver.com/merchant/cgi-bin/script_c.cgi
fields3  orderref, name, securityresponseaddress, formattedamount, stauthcode,
size
```

## 3.3 Callbackurl

`callbackurl` is the variable that determines which of the scripts (if any) listed in `callback.txt` is run following a successful authorisation. The value of `callbackurl` can be set on any page (e.g. `orderpage.html`, `form.html`) but it is the value of `callbackurl` on `form.html` that determines which script is called. For example:

- If the `form.html` value of `callbackurl` is 1, then **callback** is enabled and script 1 from `callback.txt` is called.
- If the `form.html` value of `callbackurl` is 6, then **callback** is enabled and script 6 from `callback.txt` is called.
- If the `form.html` value of `callbackurl` is null (" ") or the `callbackurl` variable is omitted altogether, then **callback** is disabled, and `success.html` is displayed as usual.

By default, `form.html` simply passes on the value of `callbackurl` that has been sent to it from either `orderpage.html` or from the shopping cart software (if you are using it). This is done by the line:

```
<INPUT TYPE=hidden NAME="callbackurl" VALUE="$callbackurl">
```

which is present in the standard template for `form.html`. So, you need to set the value of `callbackurl` in `orderpage.html` on your merchant web server, or by configuring your shopping cart software correctly.

Information on how to configure shopping carts is available in the individual shopping cart guides available from:

<http://www.securetrading.com/index.cfm?alias=downloads>

If you are not using a shopping cart, then you need to set the value of `callbackurl` by including it as a hidden field in `orderpage.html` (or its equivalent). You can do this by adding the following line inside the "form" section of `orderpage.html`:

```
<INPUT TYPE=hidden NAME="callbackurl" VALUE="1">
```

(replace 1 with 2, 3, 4... to call script 2, script 3, script 4... instead of script 1).

---

If this line is missing, or `callbackurl` is null – that is, the line reads:

```
<INPUT TYPE=hidden NAME="callbackurl" VALUE=" ">
```

Then **callback** is disabled.

### 3.4 failureurl

`failureurl` is the variable that determines which of the scripts (if any) listed in `callback-f.txt` is run following an unsuccessful authorisation. It is used in exactly the same way as `callbackurl`, so to enable it you need a line such as:

```
<INPUT TYPE=hidden NAME="failureurl" VALUE="1">
```

in your `orderpage.html` file. The default `form.html` will pass on any value of `failureurl` that it receives.

## 4 Example of callback

**Note 1:** The script `testcallback.cgi` used in this example is available for download from the SecureTrading website at:

<http://www.securetrading.com/index.cfm?alias=dl-cback>

It is part of the zip file `callback.zip`, which also contains the `callback.txt` file necessary to call the script.

**Note 2:** This script is intended for illustrative purposes only. While it may be used on your site, neither SecureTrading nor its staff can be held liable for any problems this script may cause.

This example uses **callback** to run a perl script `testcallback.cgi` on your merchant web server. We will assume that the merchant web server is configured correctly to run perl scripts, and that the script can be accessed through the URL:

<http://www.merchantwebserver.com/merchant/cgi-bin/testcallback.cgi>.

We will also assume that the order page on your merchant web server- that is, the page the customer sees immediately before receiving `form.html` – is called `orderpage.html`.

The script will produce a log file `call.log` on the merchant web server, which contains a summary of the details of each order successfully placed with the merchant – in particular, the merchant's order reference, the name and address of the customer, the value of the order, the time the order was placed, and SecureTrading's reference number for the transaction, and the authorisation code received from the bank.

### 4.1 Installing the script

You will need to upload `testcallback.cgi` to the appropriate location on your merchant web server, and create an empty file called `call.log` to hold the logged data. Make sure that these files have the relevant permissions. In this case, `testcallback.cgi` should be set to be executable, and `call.log` should have write access.

If you are unsure how to do this, please contact your web host provider or your Internet service provider (ISP). Unfortunately, SecureTrading cannot resolve such problems, since they are not based on the SecureTrading secure server.

### 4.2 callback.txt setup

Using a text editor, create a file which reads as follows:

```
method1 POST
url1 http://www.merchantwebserver.com/merchant/cgi-bin/testcallback.cgi
fields1 orderref, name, address, town, county, postcode, formattedamount,
        timestamp, streference, stauthcode
```

(Note: due to the limitations of page width, line 3 of this file has wrapped onto two lines. In your text editor, it should be typed as one line).

Remember: you should type a **single tab character** between the variable (`method1`, `url1`, etc) and the data which follows it, and you should separate the field names on line 3 using a comma and a single space.

The purpose of each line in the file is as follows:

- The content of the variable `method1` is the same as “method” in a HTML form, and normally has the value `POST`.
- The contents of the variable `url1` should be the URL of your script on your server, in this case, the URL of `testcallback.cgi` on your server.
- The contents of the variable `fields1` are the fields that are given to the script. In this example, they are the fields that the script will add to the `call.log` file on the merchant server. This example includes fields whose contents have been generated from several sources:

<code>orderref</code>	A field defined and generated by the merchant and POSTed to the SecureTrading secure server from <code>orderpage.html</code>
<code>name, address, town, county, postcode</code>	Fields entered by the customer and POSTed to the SecureTrading secure server from <code>orderpage.html</code>
<code>streference</code>	A field containing a unique code generated by the SecureTrading secure server to identify the transaction
<code>formattedamount</code>	A field generated by the SecureTrading secure server from data originally POSTed to the SecureTrading secure server from <code>orderpage.html</code>
<code>stauthcode</code>	A field generated by the bank server

For more information about these fields, together with a full list of all the fields available, see the reference section of the **ST Payment Pages setup guide** (note that you can modify the list of fields to be logged, and the script should still work).

Once you have created `callback.txt`, you should upload it to the SecureTrading secure server using the upload facility of **My-ST**.

### 4.3 callbackurl setup

Edit `orderpage.html` on your merchant web server and add the following line:

```
<INPUT TYPE=hidden NAME="callbackurl" VALUE="1">
```

Save the changes.

### 4.4 Testing

**Be careful:** Testing can only be done while your SecureTrading account is in test mode. All new SecureTrading accounts are in test mode by default, and remain so until you request to switch to live mode.

You can test using the following dummy credit card details:

To test for correct operation, browse to your e-commerce site using a web browser. Put together a dummy order, enter the appropriate details on `orderpage.html`, and progress to `form.html`. Now use the following credit card details:

Credit card type: Visa  
Credit card number: 4111 1111 1111 1111  
If Switch, issue number: leave blank  
Expires end: 05/10

Click *submit* and wait for the result.

You should find that `call.log` has been updated with the fields and their values as governed by `fields1` in `callback.txt`. If not, check the [Troubleshooting](#) section at the end of this guide.

If you want to test the operation of your system when a transaction is declined, use the following credit card details:

Credit card type: Visa  
Credit card number: 4242 4242 4242 4242  
If Switch, issue number: leave blank  
Expires end: 05/10

In this example, you should observe the default behaviour of the system (`failure.html` displayed, `failureemail.txt` sent to merchant).

## 4.5 Logging failed authorisations

If you want to log failed authorisations as well as successful ones, you need to run **callback** following an unsuccessful authorisation, as well as after a successful authorisation. To do this:

- Create a `callback-f.txt` file which reads as follows:

```
method1 POST
url1 http://www.merchantwebserver.com/merchant/cgi-bin/
fields1 orderref, name, address, town, county, postcode,
formattedamount, timestamp, streference, stauthcode
```

(Note that in this case, the same script `testcallback.cgi` is run whether the transaction is authorised or not. In general, you can run the same script or different scripts depending on the results of authorisation – there are no restrictions).

- Upload `callback-f.txt` to the SecureTrading secure server.
- Edit `orderpage.html` on your merchant web server and add the following line:

```
<INPUT TYPE=hidden NAME="failureurl" VALUE="1">
```

- Save the changes.

`call.log` will now contain details of unsuccessful as well as successful transactions. You can test this by using the 4242... credit card details given above.

## 5 Further customisation

You can add a couple of extra variables to `callback.txt` and `callback-f.txt` to fine-tune the way that **callback** behaves.

### 5.1 pipe1, pipe2, etc

`pipeX` (where X is a number from 1 to 8) is an optional variable that governs whether `success.html` (or `failure.html`) is displayed to the customer after a transaction. It can take the value `yes` or `no`, and is inserted in `callback.txt` (or `callback-f.txt`) like this:

```
method1 POST
url1      http://www.merchantwebserver.com/merchant-cgi-bin/successscript.cgi
fields1   orderref, name, address, postcode, formattedamount, stauthcode
pipe1     yes
```

(Note: tab character between `pipe1` and `yes`, not spaces).

If `pipeX` is set to `yes`, then `success.html` (or `failure.html`) is **not** displayed following a successful transaction. Instead, what your customer sees is governed by commands in your script. In the `testcallback.cgi` example, your customer sees “*Starting...Finished*”.

If `pipeX` is set to `no`, then the customer sees `success.html` (or `failure.html`), although the script still runs.

If you are using multiple scripts, then use `pipe2`, `pipe3` etc to control the behaviour when script 2, script 3 etc are run:

```
method1 POST
url1      http://www.merchantwebserver.com/merchant-cgi-bin/script_a.cgi
fields1   orderref, name, address, postcode, formattedamount, stauthcode
pipe1     yes
method2 POST
url2      http://www.merchantwebserver.com/merchant-cgi-bin/script_b.cgi
fields2   orderref, name, address, postcode, formattedamount, stauthcode, color
pipe2     no
method3 POST
url3      http://www.merchantwebserver.com/merchant-cgi-bin/script_c.cgi
fields3   orderref, name, address, postcode, formattedamount, stauthcode, size
pipe3     yes
```

**Note:** `pipeX` does **not** control the sending of merchant customer emails following a transaction. Instead, this is controlled by the `merchantemail` and `customeremail` variables – see the **ST Payment Pages setup guide** for more information.

### 5.2 email1, email2 etc

`emailX` (where X is a number from 1 to 8) is an optional variable that controls whether the output of the script is also sent to the merchant, as an email. Its value is the email address that the information should be sent to, and it is inserted in `callback.txt` (or `callback-f.txt`) like this:

```
method1 POST
url1      http://www.merchantwebserver.com/merchant-cgi-bin/successscript.cgi
fields1   orderref, name, address, postcode, formattedamount, stauthcode
email1    John.Smith@scalemodelcars.co.uk
```

The email contains the name of the script called, and also the output sent by the script to the customer’s browser. For example, if you used the email variable in the example given in [Example of callback](#), then your email would look like:

---

Your cgi script :  
`http://www.merchantwebserver.com/merchant/cgi-bin/testcallback.cgi`  
was called from within your SecureTrading secure form.

The output of this script was:

```
HTTP:/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Tue, 11 Apr 2000 14:48:43 GMT
Content-type: text/html
```

```
<html><body><br></body></html>
StartingFinished
```

This feature was designed to help you debug your scripts while your SecureTrading account is in test mode, and is not recommended for use in live mode.

---

## 6 Using callback

**Callback** is a versatile feature which gives you access to information entered by your customer, information generated by the SecureTrading secure server, information received from the bank server and information generated by your merchant web server.

However, it is up to you what you do with this information; SecureTrading cannot supply or tailor scripts for your own use. If you need help writing the script necessary for your system, you should seek assistance from a web developer.

SecureTrading's technical support cannot debug merchants' scripts, so please do **not** contact us if you have such a problem.

---

## 7 Troubleshooting

- Check that the URL of the script is correct
- Check that the files used by your script (if any) have the correct permissions.
- Check that your script file has the executable permission set.
- Check that the value of `callbackurl` passed through `form.html` is 1 (or 2, 3, 4... to run script 2, script 3, script 4...). You can do this by temporarily adding a line to display this value on `form.html`.
- Check the correct versions of `callback.txt` and `callback-f.txt` are uploaded to the SecureTrading secure server.