# Generating Mass Keys

July 2018

CyberSource®
the power of payment

CyberSource Corporation HQ | P.O. Box 8999 | San Francisco, CA 94128-8999 | Phone: 800-530-9095

## CyberSource Contact Information

For general information about our company, products, and services, go to http://www.cybersource.com.

For sales questions about any CyberSource Service, email sales@cybersource.com or call 650-432-7350 or 888-330-2300 (toll free in the United States).

For support information about any CyberSource Service, visit the Support Center: http://www.cybersource.com/support

## Copyright



## Restricted Rights Legends



## Trademarks

# Contents

# Recent Revisions to This Document

| Release | Changes |
|---|---|
| July 2018 | Initial release. |

# About This Guide

## Audience and Purpose

This guide is for portfolio and account-level users or developers who want to mass-generate security keys for their MIDs where their integration method would be:

- Simple Order API
- SCMP API
- SOAP Toolkit API
- Shared Secret Key

## Conventions

### Note and Important Statements

**Note**

A *Note* contains helpful suggestions or references to material not contained in the document.

**Important**

An *Important* statement contains information essential to successfully completing a task or learning a concept.

## Text and Command Conventions

| Convention | Usage |
|---|---|
| **Bold** | ■ Field and service names in text; for example: Include the **ics_applications** field.<br>■ Items that you are instructed to act upon; for example: Click **Save**. |
| `Screen text` | ■ XML elements.<br>■ Code examples and samples.<br>■ Text that you enter in an API environment; for example: Set the **davService_run** field to `true`. |

# Related Documents

- *CyberSource REST API Getting Started Guide* (PDF | HTML)
- *CyerSource REST API Reference* (HTML)
- *Universal Management Portal User Guide* (PDF | HTML)

Refer to the Support Center for complete CyberSource technical documentation:

http://www.cybersource.com/support_center/support_documentation

# Customer Support

For support information about any CyberSource service, visit the Support Center:

http://www.cybersource.com/support

CHAPTER 1

# Generating Keys Using the SDK

In order to send a request to the CyberSource API, you need a security key. This software developer kit (SDK) enables portfolio users to mass generate security keys for their MIDs. The process consists of six main steps:

1. Create your merchants in the Universal Management Portal. For details on creating merchants, see the *Universal Management Portal User Guide*.
2. Create a p12 security key for your account in order to access the SDK/API. See "Creating a Security Key," page 8.
3. Download the Mass Key Generation SDK from:

   https://www.cybersource.com/developers/integration_methods/simple_order_and_soap_toolkit_api/
4. Update the *configuration.properties* file. See "Updating the Configuration File," page 9.
5. Create a *.csv* file with the list of merchants for whom you want to generate keys. See "Creating the Merchant List CSV File," page 10.
6. Run *startup.sh* (for Linux) or *startup.bat* (for Windows).

**Important** Java 1.8 and higher is required in order to run the SDK. You must also have the Java Cryptography Extension (JCE) installed. For details on downloading the software and setting up your environment:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

# Creating a Security Key

Before you can mass-generate keys for your MIDs, you must generate your own security key. You can use either a certificate key or a shared secret key.

## To create a Certificate security key:

**Step 1** Log in to the Universal Management Portal.

**Step 2** From the left navigation panel, choose **Portfolio Tools > Key Management**. You are redirected to the Key Management page in the New Business Center.

**Step 3** From the Keys Type drop-down menu, choose **API Keys**.

**Step 4** Click the **+** icon at the bottom of the page to generate a new key.

**Step 5** In the Generate Key panel, choose **Certifcate**.

**Step 6** Click **Download**.

**Step 7** When prompted, save the *certificate.jnlp* file to your system.

**Step 8** Double-click the ***.jnlp*** file to run it. The CyberSource Key Generation dialog box opens.

**Step 9** Click **Generate Certificate Request**. The Select Key Save Location dialog box opens.

**Step 10** Choose a location in which to store the .p12 key.

**Step 11** Close the CyberSource Key Generation dialog box.

## To create a Shared Secret security key:

**Step 1** Log in to the Universal Management Portal.

**Step 2** From the left navigation panel, choose **Portfolio Tools > Key Management**.

**Step 3** You are redirected to the Key Management page in the New Business Center.

**Step 4** From the Keys Type drop-down menu, choose **API Keys**.

**Step 5** Click the **+** icon at the bottom of the page to generate a new key.

**Step 6** In the Generate Key panel, choose **Shared Secret**.

**Step 7** Click **Generate New Key**.

**Step 8** Click **Download**.

**Step 9** When prompted, save the *certificate.txt* file to your system.

**Step 10** Close the CyberSource Key Generation dialog box.

# Updating the Configuration File

The SDK contains a *configuration.properties* file, which contains the information needed to mass generate security keys for your MIDs. You must edit the following fields:

**Table 1 Fields in the configuration.properties File**

| Field Name | Definition |
|---|---|
| partnerId | Organization ID of the partner. |
| partnerKey | The absolute path to the location where the partner *.p12* or *.txt* key is stored. This is the location you selected in "Creating a Security Key." |
| keyAlias | Alias for partner *.p12* key. |
| keyPassword | Password for partner *.p12* key. |
| action | The action that you want performed when you run the SDK. The options are:<br>■ create<br>■ deactivate |
| keyType | The type of security key that you want to create. The options are:<br>■ Simple Order API<br>■ SOAP Toolkit API<br>■ SCMP API<br>■ Shared Secret |
| merchantListLocation | The location of your *.csv* file containing your list of MIDs. For more information about creating this file, see "Creating the Merchant List CSV File," page 10. |
| resultFolderLocation | The location where the generated keys should be stored. |

# Creating the Merchant List CSV File

When you run the SDK, you must provide a CSV file that contains a list of MIDs for which you want to generate security keys. The header of the CSV file must contain the following fields:

- Reference Number
- MID
- Serial Number

**Example Merchant List Input File**

```
Reference Num, MID, Serial Number

1001, mid1,
1002, mid2,
1003, mid3,
```

# Viewing Output Results

When the security keys are successfully generated, you can view the output in the folder that you specified in the configuration file. The output type is dependent on the API key type that you selected.

## Simple Order API

When you run the *startup.sh*/*startup.bat* file, the security keys are generated and certified on the client machine, and the public certificate is registered with CyberSource. For each MID in your CSV file, a *.p12* key is created and stored in the result folder that you specified in the configuration.propteries file. A *.csv* file is also created in the results folder and contains the following information:

- Reference Number
- MID
- Serial Number
- Expiration Date
- Status
- Message

## SOAP Toolkit API

When you run the *startup.sh*/*startup.bat* file, the security keys are generated and certified on the client machine, and the public certificate is registered with CyberSource. The keys are stored in the logs folder within the results folder that you specified in the *configuration.properties* file. A *.csv* file is also created in the results folder and contains the following information:

- Reference Number
- MID
- Serial Number
- Expiration Date
- Status
- Message

## SCMP API

When you run the *startup.sh*/*startup.bat* file, the security keys are generated and certified on the client machine, and the public certificate is registered with CyberSource. The keys (*.pvt* and *.crt*) are stored in a folder within the results folder that you specified in the *configuration.properties* file. A *.csv* file is also created in the results folder and contains the following information:

- Reference Number
- MID
- Serial Number
- Expiration Date
- Status
- Message

## Shared Secret

When you run the *startup.sh*/*startup.bat* file, the security keys are generated and certified on the client machine, and the public certificate is registered with CyberSource. The keys are stored in the logs folder within the results folder that you specified in the *configuration.properties* file. A *.csv* file is also created in the results folder and contains the following information:

- Reference Number
- MID
- Serial Number
- Expiration Date
- Status
- Message

CHAPTER 2

# Generating Keys Using the API

You can use the Mass Key Generation REST API to create or delete multiple keys simultaneously. For registration and authentication information, see the *CyberSource REST API Getting Started Guide* (PDF | HTML).

## Creating Keys

### Request

Operation and path—`POST /v1/soap-keys/`

Server URL: `https://api.cybersource.com/kms/v1/soap-keys/`

**Table 2 Fields in the Create Keys Request Payload**

| Field Name | Data Type | Description | Required/ Optional |
|---|---|---|---|
| clientReferenceInformation | Object | Contains reference information about the client. | Optional |
| ▪ code | String | Client-generated order reference or tracking number. CyberSource recommends that you send a unique value for each transaction so that you can perform meaningful searches for the transaction. | Optional |
| keyInformation | Array of object key information | Contains information about this key. | Required |
| ▪ organizationId | String | Merchant ID of the account submitting the request. | Required |
| ▪ referenceNumber | String | Key reference number. | Required |

**Example 1 Request Payload**

Authorization: Bearer [encrypted string]

```
{
   "clientReferenceInformation": {
      "code": "string"
   },
   "keyInformation": [{
      "organizationId": "string",
      "referenceNumber": "string"
   }]
}
```

# Responses

**Table 3 Fields in a 201 Successful Response**

| Field Name | Data Type | Description |
|---|---|---|
| id | String <= 26 characters | A unique identification number assigned by CyberSource to identify the submitted request. |
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |
| status=”ACCEPTED” | String | The status of the submitted transaction. |
| clientReferenceInformation | Object | Contains reference information about the client. |
| ■ code | String | Client-generated order reference or tracking number. CyberSource recommends that you send a unique value for each transaction so that you can perform meaningful searches for the transaction. |
| keyInformation | Object | Contains information about this key. |
| ■ organizationId | String | Merchant ID of the account that submitted the request. |
| ■ referenceNumber | String | Key reference number. |
| ■ serialNumber | String | Key serial number. |
| ■ key | String | The value of the key. |

**Table 3 Fields in a 201 Successful Response (Continued)**

| Field Name | Data Type | Description |
|---|---|---|
| ■ status | String | The status of the key. |
| ■ expirationDate | String | Valid values are SUCCESS or FAILED. |
| ■ message | String | Detailed message related to the status and reason. |

**Example 2 201 Response**

```
{
  "id": "string",
  "submitTimeUtc": "string",
  "status": "ACCEPTED",
  "clientReferenceInformation": {
    "code": "string"
  },
  "keyInformation": [
    {
      "organizationId": "string",
      "referenceNumber": "string",
      "serialNumber": "string",
      "key": "string",
      "status": "SUCCESS",
      "expirationDate": "string",
      "message": "string"
    }
  ]
}
```

**Table 4 Fields in a 400 Invalid Request Response**

| Field Name | Data Type | Description |
|---|---|---|
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |
| status="INVALID_REQUEST" | String | The status of the submitted transaction. |

**Table 4 Fields in a 400 Invalid Request Response (Continued)**

| Field Name | Data Type | Description |
|---|---|---|
| reason | String | Reason for the status.<br>Valid values include:<br>■ MISSING_FIELD<br>■ INVALID_DATA<br>■ DUPLICATE_REQUEST |
| message | String | Detailed message related to the status and reason. |
| details | Array of object | More information about the error. |
| ■ field | String | The flattened JSON object field name/ path that is either missing or invalid. |
| ■ reason | String | Possible reasons for the error.<br>Valid values include:<br>■ MISSING_FIELD<br>■ INVALID_DATA |

**Example 3 400 Response**

```
{
  "submitTimeUtc": "string",
  "status": "INVALID_REQUEST",
  "reason": "MISSING_FIELD",
  "message": "string",
  "details": [
    {
      "field": "string",
      "reason": "MISSING_FIELD"
    }
  ]
}
```

**Table 5 Fields in a 502 Response**

| Field Name | Data Type | Description |
|---|---|---|
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |

**Table 5 Fields in a 502 Response (Continued)**

| Field Name | Data Type | Description |
|---|---|---|
| status="SERVER_ERROR" | String | The status of the submitted transaction. |
| reason | String | Reason for the status.<br>Valid values include:<br>■ SYSTEM_ERROR<br>■ SERVER_TIMEOUT<br>■ SERVICE_TIMEOUT |
| message | String | Detailed message related to the status and reason. |

**Example 4 502 Response**

```
{
  "submitTimeUtc": "string",
  "status": "SERVER_ERROR",
  "reason": "SYSTEM_ERROR",
  "message": "string"
}
```

# Deleting Keys

## Request

Operation and path—`POST /v1/soap-keys/deletes/`

Server URL: `https://api.cybersource.com/kms/v1/soap-keys/deletes/`

**Table 6 Fields in the Delete Keys Request Payload**

| Field Name | Data Type | Description | Required/ Optional |
|---|---|---|---|
| clientReferenceInformation | Object | Contains reference information about the client. | Optional |

**Table 6 Fields in the Delete Keys Request Payload (Continued)**

| Field Name | Data Type | Description | Required/ Optional |
|---|---|---|---|
| ■ code | String | Client-generated order reference or tracking number. CyberSource recommends that you send a unique value for each transaction so that you can perform meaningful searches for the transaction. | Optional |
| keyInformation | Array of object key information | Contains information about the key that is to be deleted. | Required |
| ■ organizationId | String | Merchant ID of the account submitting the request. | Required |
| ■ referenceNumber | String | Key reference number. | Required |
| ■ serialNumber | String | Key serial number. | String |

**Example 5 Request Payload**

```
{
  "clientReferenceInformation": {
    "code": "string"
  },
  "keyInformation": [
    {
      "organizationId": "string",
      "referenceNumber": "string",
      "serialNumber": "string"
    }
  ]
}
```

# Responses

**Table 7 Fields in a 200 Response**

| Field Name | Data Type | Description |
|---|---|---|
| id | String <= 26 characters | A unique identification number assigned by CyberSource to identify the submitted request. |

**Table 7 Fields in a 200 Response (Continued)**

| Field Name | Data Type | Description |
|---|---|---|
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |
| status=”ACCEPTED” | String | The status of the submitted transaction. |
| clientReferenceInformation | Object | Contains reference information about the client. |
| ■ code | String | Client-generated order reference or tracking number. CyberSource recommends that you send a unique value for each transaction so that you can perform meaningful searches for the transaction. |
| keyInformation | Object | Contains information about this key. |
| ■ organizationId | String | Merchant ID of the account that submitted the request. |
| ■ referenceNumber | String | Key reference number. |
| ■ serialNumber | String | Key serial number. |
| ■ status | String | The status of the key. |
| ■ message | String | Detailed message related to the status and reason. |

**Example 6 200 Response**

```
{
  "id": "string",
  "submitTimeUtc": "string",
  "status": "ACCEPTED",
  "clientReferenceInformation": {
    "code": "string"
  },
  "keyInformation": [
    {
      "organizationId": "string",
      "referenceNumber": "string",
      "serialNumber": "string",
      "status": "SUCCESS",
      "message": "string"
    }
  ]
}
```

**Table 8 Fields in a 400 Response**

| Field Name | Data Type | Description |
|---|---|---|
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |
| status="SERVER_ERROR" | String | The status of the submitted transaction. |
| reason | String | Reason for the status.<br>Valid values include:<br>■ MISSING_FIELD<br>■ INVALID_DATA<br>■ DUPLICATE_REQUEST |
| message | String | Detailed message related to the status and reason. |
| details | Array of object | More information about the error. |
| ■ field | String | This is the flattened JSON object field name/path that is either missing or invalid. |
| ■ reason | String | Possible reasons for the error.<br>Valid values include:<br>■ MISSING_FIELD<br>■ INVALID_DATA |

**Example 7 400 Response**

```
{
  "submitTimeUtc": "string",
  "status": "INVALID_REQUEST",
  "reason": "MISSING_FIELD",
  "message": "string",
  "details": [
    {
      "field": "string",
      "reason": "MISSING_FIELD"
    }
  ]
}
```

**Table 9 Fields in a 502 Unexpected Error or System Timeout Response**

| Field Name | Data Type | Description |
|---|---|---|
| submitTimeUtc | String | Time of request in UTC.<br>Format: YYYY-MM-DDThh:mm:ssZ<br>For example, 2016-08-11T22:47:57Z equals August 11, 2016, at 22:47:57 (10:47:57 p.m.). The T separates the date and the time. The Z indicates UTC. |
| status="SERVER_ERROR" | String | The status of the submitted transaction. |
| reason | String | Reason for the status.<br>Valid values include:<br>■ SYSTEM_ERROR<br>■ SERVER_TIMEOUT<br>■ SERVICE_TIMEOUT |
| message | String | Detailed message related to the status and reason. |

**Example 8 502 Response**

```
{
  "submitTimeUtc": "string",
  "status": "SERVER_ERROR",
  "reason": "SYSTEM_ERROR",
  "message": "string"
}
```