GP webpay - Private Key Management and signing messages

Version: 1.3 Global Payments Europe, s.r.o. Created 19.2.2016 Last update 6.12.2019

8

SERVICE. DRIVEN. COMMERCE

globalpaymentsinc.com

Author	GPE Application Development
Manager	
Approved	
Version	1.3
Confidentiality	Confidential

Document history:

Version	Date	Author	Comments	
1.0	19.2.2016	GPE Application Development	Initial document version	
1.1	29.3.2016	GPE Application Development	Corrections	
1.2	3.5.2019	GPE Application Development	Merged with "GP_webpay_Signing_messages_v1.0_EN.docx"	
1.3	25.11.2019	GPE Application Development	New version of the GP webpay Kestore Manager Examples update	

Table of contents

1.	Formula	clause	4
2.	Introduc	tion	5
2	.1 Ger	neral GP webpay security principle	5
	2.1.1	How to obtain the private key	5
	2.1.2	PKI functions	5
2	.2 Usi	ng PKI in GP webpay	6
	2.2.1	Ways of use	6
	2.2.2	Message integrity verification	6
	2.2.3	Message sender identity verification	6
3.	Private I	key and its management	8
3	5.1 Priv	ate key in general	8
З	.2 Hov	v to obtain the private key	8
	3.2.1	History	8
	3.2.2	The present	9
3	.3 The	e private key management	. 10
	3.3.1	Public key information	.11
	3.3.2	Format update	.14
	3.3.3	Password change	.18
	3.3.4	For developers	. 21
4.	Signing	messages	. 25
2	.1 Ger	neral technical basis	. 25
	4.1.1	Signing a request	.25
	4.1.Z	Response verification	. 26
	4.1.3 4.1.4	Verification of the electronic signature	. 20
	4.1.5	Graphic representation of key generation and verification.	.28
	4.1.6	Kevs used	. 28
	4.1.7	Logging	. 29
	4.1.8	References	. 29
Z	.2 Dig	est examples	. 30
	4.2.1	Test key and application ZIPs files	. 30
	4.2.2	Digest example	. 34



1. Formula clause

This document including any possible annexes and links is intended solely for the needs of an eshop service provider (hereinafter referred to as "Customer").

Information included in this document (hereinafter referred to as "Information") are subject to intellectual property and copyright protection of the Global Payments Europe, s.r.o. (hereinafter referred to as "GPE") and are of a commercially confidential nature in accordance with the provisions of the section 504 of the Act No. 89/2012 Coll., Civil Code. The Customer is aware of the legal obligations in relation to the handling of Information.

Information or any part thereof may not be provided or in any way made available to third parties without the prior written consent of the GPE. At the same time, Information may not be used by the Customer for purposes other than for the purpose for which it serves. To avoid any doubts, without the prior written consent of the GPE, Information or any part thereof may be provided or in any way made available neither to companies providing payment processing services on the Internet.

The GPE to the extent permitted by applicable law retains all rights to this document and Information contained therein. Any reproduction, use, exposure, or other publication, or dissemination of Information or its part by methods known and as yet undiscovered without the prior written consent of the GPE is strictly prohibited. The GPE is not in any way responsible for any errors or omissions in Information. GPE reserves the right, without giving any reason, to amend or repeal any Information.

2. Introduction

This document describes principle of creating payments in the GP webpay payment gateway environment and authorization of subsequent operations with payments.

2.1 General GP webpay security principle

In order to secure itself, the GP webpay system uses the so-called PKI (Public Key Infrastructure) model. This model uses asymmetric cryptography, using two different keys.

- 1. Private key this part is secret and is owned only by the authorized person (i.e. key owner)
- 2. Public key the public part that can be distributed any channel (even the unsecured) e-mail, public repository of keys ...

A main characteristic of the private key is that no two keys in the world are identical – i.e. each and every key is original.

2.1.1 How to obtain the private key

- Public certification authority a trusted commercial authority providing management of keys (i.e. issuance, revocation, renewal...). Its public key is located directly in web browsers, or in various run-times (run-time environment for other software e.g. Java, .NET...). Keys issued by this authority are widely accepted as credible and are used for communication with banks and public institutions e.g. Thawte (https://www.thawte.com/), První certifikační autorita a.s. (http://www.ica.cz/).
- Various general solutions private keys are not generally accepted, nevertheless they are built on trust between a client and specific key provider – e.g. Komerční banka has its own certification authority and provides its clients with keys to enable their communication with internet banking.
- GP webpay enables its clients to obtain a private key via the web portal. This key can be used only in the GP webpay environment.

2.1.2 PKI functions

- access authentication (user's identity verification)
- messages integrity verification (message has not been altered in any way)
- undeniableness using electronic signature
- privacy messages encryption, symmetric and asymmetric encryption

GP webpay uses only two of the above mentioned functions – integrity verification and undeniableness.

2.2 Using PKI in GP webpay

2.2.1 Ways of use

The private key is used for generating a signature of all the messages enabling manipulation with payments. A verified signature guarantees integrity of transmitted data and correct identity (undeniableness of identity) of a message sender – there is no possibility to create a signature using the public part of key.

Types of messages:

- Creating new payments by means of standard HTTP interface
- Payments management in the Portal money capture/refund of a card holder
- Payments management by means of web-services used at direct connection of the GP webpay system with merchant's payment system

2.2.2 Message integrity verification

Every message modifying data (whether it is creating payments or operations with payments) contains not only data, but also a signature box. Signature is created on the side of the private key owner on the basis of input data and the private key with the use of a general algorithm for signature calculation. After the data are sent to the server, it carries out verification a similar way, but using relevant public key (public key is searched out on the basis of identifier of the message sender; the identifier is sent in a message). If the calculation differs, there must have occurred data disruption in the course of their transmission.



2.2.3 Message sender identity verification

A part of transmitted data is also an identifier of the message sender. On the basis of this identifier, the relevant public key is selected on the server side. If it was possible to verify the signature and provided that there are no two identical private keys, it can be stated that the data had really been sent by the private key holder.



3. Private key and its management

3.1 Private key in general

Private key is the basis of security of the GP webpay system. This key is solely owned by the key holder and it is necessary to observe maximally all the security requirements for confidentiality:

- Keep it in a secure place
- Always have it protected by password
- If the key is compromised, it is necessary to obtain a new key and to inform all the subjects using its public part for identity verification that the key has been compromised

Private key is stored in a data file. This file is called a repository, or keystore. Keystore can contain more private and public keys. To be able to differentiate among the keys in the keystore, there are names assigned to them – the so-called aliases. The keystore is protected by the central password and besides that, every private key is protected by its own password.

There are several formats of keystores. For our purposes, the following ones are sufficient (the below described conversion application supports precisely these formats):

JCEKS - keystore in the format supported by JAVA programming language

PFX – keystore in the format supported by Microsoft corporation (PKCS12)

PEM – keystore in the format supported by PHP programming language

The formats for distribution of the public key relate to these types as well:

PEM - keystore in the text format

DER - keystore in the binary format

3.2 How to obtain the private key

As mentioned above, private key can be obtained in a few different ways. For commercial purposes, or for communication with public administration, it is necessary to obtain a key from a trusted certification authority.

For the purposes of operating the GP webpay system, it is sufficient to obtain it by means available in the GP webpay.

If you have already bought any private key (there are several commercial certification authorities issuing/selling private keys), it can be used as well.

3.2.1 History

From the very beginning of functioning of the GP webpay, there has been a possibility to get the private key by means of individually delivered application "Key and certificate generation". This application has been downloadable from the user environment GP webpay GUI and also as a part of distribution package of the documentation.

🛓 Certificate and keystore generation	
Company/Merchant name:	
Merchant Number (Merchant ID):	
Certificate sequence number:	201605021535
Merchant bank:	< select bank> 💌
Keystore file name:	
	(.ks, .cer, .pfx, .pem files will be generated)
Keytore password (7 letters/numbers max.):	
	(same will be use for private key access)
(c) 2004/2009 GPE, s.r.o.	Generate cert/keystore

The following files are the result of generation:

<name>.ks - keystore file in the Java format - contains both the private and the public key

<name>.pfx - keystore file in the PKCS#12 format - contains both the private and the public key

<name>.pem - keystore file in the PEM format - contains both the private and the public key - e.g. for PHP apps

<name>.cer - file containing the public key

3.2.2 The present

The GP webpay Portal, the new graphic interface for management of orders, has incorporated management of both private and public keys of individual e-shops. Its part is also the possibility to generate the private key by means of the web browser.

$\langle \rangle$	් KEY MANAGEMENT	
	Creating keys and their management The GP webpay payment gateway uses principles of asymmetric cryptography to ensure the security based on private and public keys. The private key will be stored in the web browser's cache memory and the public key will be stored on server side of the GP webpay. The user has the ability to manage the payment orders through the menu Payments (for managing payment orders it's always required to provide a password for the private encryption key). After selection of the e-shop from the list, there is displayed the scheme containing information about the private key status in the web browser and information about the public key status on the server side of the GP webpay. These keys are also used to secure communication between the e-shop and the GP webpay server and also at using the Web Services. Learn more.	Creating a private key t creates securely a new private key in your web browser. This option is intended for users who do not own any private keys. CREATE
© Global Pa	- ayments Europe 2015, v: 1.2.2-S	🛞 English

The result of generation is the file "gpwebpay-pvk.key" in text format PEM.

This key can be inserted into the web browser (by import in the Portal).

At the same time it has to be preserved for future use - e.g. in another web browser.

3.3 The private key management

To be able to operate with payments in the web application GP webpay Portal (hereinafter referred as the Portal), it is necessary to upload the private key to the web browser. This upload can be carried out after a successful login, directly in the environment of the Portal. The private key has to be stored in the text format PEM; this format is created also during key generation in the Portal (file "gpwebpay-pvk.key").

In case you have the private key from earlier, it is necessary to update the original format to the new one. The GP webpay Keystore Manager application can be used for this purpose. Application is available in the "Downloads" menu of the Portal and for its operation it requires to have the installed Java (downloadable from Oracle website <u>http://www.java.com</u>).

GP webpay Keystore Manager application has these functionalities:

- Public key information the key "fingerprint" it can be compared with the value in the GP webpay Portal
- Format update format conversion of the initial file containing the private key
- Password change changing the private key password in the new format
- For developers automatic conversion of the private key in the new format into formats supported by various developer tools



Hover your mouse over a "tile" to display a brief description of functionality:

GP webpay Keystore Manage	r			
ECOMMERCE GP WEBPAY				
View of the shortened vers key fingerprint INFORMATION ABOUT PUBLIC KEY	sion of the public The pr use wi FORMAT UPDAT FORMAT UPDAT	ivate key format has to be updated for thin the Portal GP webpay. CHANGF CHANGF Changing the private	DEVELOPERS key password Transferring the formats for integra into e-shop.	private key into different ating the payment gateway
© Global Payments Europe			* 💶 🗖 📕	

The application supports a few language versions. To switch among them, use the flag icons in the bottom part of the screen.

3.3.1 Public key information

"Tile" is designed to show the "fingerprint" of the public key. The "Fingerprint" can be compared with value in GP webpay Portal to be sure that the key in keystore is the same as the key stored GP webpay server.

By clicking the "INFORMATION ABOUT PUBLIC KEY" tile, there is displayed a window to select the file containing the private key:

💈 GP web	GP webpay Keystore Manager					
g	ECOMMERCE GP WEBPAY					
		Select the private key file				
	File path		Browse			
	Password Pa	assword				
5	Select the keystore with t (he extension (.ks or .jks) which has to be conv .key) for the use within the Portal GP webpay.	erted to the new format			
		CA	NCEL			

In the input box "File path", it is necessary to find the file containing the original key by scrolling through the directory structure.

	ſ	Otevřít							x
			očítač 🕨	OS (C:) ► klic	▼ \$ 9	Prohledat: k	dic		Q
		Uspořádat 🔻 🕴	lová slož	ka			•== •		0
GP webpay Keystore Manager		🝊 OneDrive	^	Název položky	Datum změny	Тур	Velikos	st	
	E	Chihovny		条 pay-test.ks 윢 pay-test-vice.ks	28.8.2008 13:20 4.6.2015 12:07	Soubor KS Soubor KS		3 kB 18 kB	
File path	Select a priva	 Obrazky Subversion Videa Počítač 	II						
File path		🚢 OS (C:)							
Password Select the keystore wit	Password h the extension (.ks or .jks) (.kev) for the use within t	G (5)	Název	souboru:	•	JKS Keystore Otevřít	files (*.ks,*	*.jks,*.jce Storno	•
		CA	NCEL	NEXT					

Confirm your selection of the private key file by clicking the "Open" button.

The selection window closes and the application waits for the password to the keystore and for pressing the "Next" button. There follows an attempt to retrieve the file content.

If the wrong password is entered, or if the file does not contain the private key, the following message is displayed:



In case that the keystore file contains more private keys, the list of keys is displayed and it is necessary to select the right private key:

💈 GP wet	bpay Keystore Manager	x
Ş	ECOMMERCE GP WEBPAY	
	Select the private key file	
paymuzo		
gpwebpa	ау	
	Select the private key which has to be updated to the new format (with the extension .key)	
PRE	EVIOUS CANCEL NEXT	

and continue by clicking the "Next" button". If there is only one private key, this screen is skipped.

After the correctness of the input file is verified, or key selection is confirmed, the "fingerprints" are displayed:

🗐 GP w	vebpay Keystore Manager					
g	ECOMMERCE GP WEBPAY					
	Information about public key					
	Fingerprint 060330d0c9bd2a465c145e0f14f2fbaa4b914f8d Short fingerprint 4B:91:4F:8D					
	The shortened version of the fingerprint corresponds to the GP webpay.					
PRE	EVIOUS	FINISH				

Displayed value should be the same as the value in GP webpay Portal:



By pressing the "Finish" button, the application returns to the initial screen.

3.3.2 Format update

This "tile" serves for the format update of the initial private key, which was used in the old GUI for merchants. The original format is in the JAVA structure and has – in most cases – file name extension ".ks", ".jks" or ".jceks". The new format is in the PEM structure and the private key is stored in the file "gpwebpay-pvk.key".

By clicking the "FORMAT UPDATE" tile, there is displayed a window to select the file containing the private key in the old format:

GP webpay Keystore Manager					
GP WEBPAY	E				
	Select the private key file				
File path		Browse			
Password	Password				
Select the keystore wi	th the extension (.ks or .jks) which has to be convert (.key) for the use within the Portal GP webpay.	ed to the new format			
	CANO	CEL NEXT			

In the input box "File path", it is necessary to find the file containing the original key by scrolling through the directory structure.

	Otevřít	— ו
	Colored C: Value C: V	 ✓ 4 Prohledat: klic
	Uspořádat 👻 Nová složka	III 🔹 🗖 🔞
GP webpay Keystore Manager	🐔 OneDrive 🔺 Název položky Datum z	rměny Typ Velikost
GP WEBPAY Select a pr	 Knihovny Dokumenty Hudba Obrázky Subversion Videa 	8 13:20 Soubor KS 3 kB 12:07 Soubor KS 18 kB
File path Password Password	Počítač	
	Název souboru:	✓ JKS Keystore files (*.ks,*.jks,*.jce ▼
Select the keystore with the extension (.ks or .j (.key) for the use with	s)	Otevřít 🔽 Storno
	CANCEL	

Confirm your selection of the private key file by clicking the "Open" button.

The selection window closes and the application waits for the password to the original keystore and for pressing the "Next" button. There follows an attempt to retrieve the file content.

If the wrong password is entered, or if the file does not contain the private key, the following message is displayed:



In case that the keystore file contains more private keys, the list of keys is displayed and it is necessary to select the right private key:

💈 GP web	opay Keystore Manager	_ D _ X	
g	ECOMMERCE GP WEBPAY		
	Select the private key file		
paymuzo			
gpwebpay	у		
Select the private key which has to be updated to the new format (with the extension .key)			
PRE	EVIOUS	NEXT	

and continue by clicking the "Next" button". If there is only one private key, this screen is skipped.

After the correctness of the input file is verified, or key selection is confirmed, you are prompted for selection of the destination directory to save the converted file and a request to enter a new password for the private key. The password has to be entered twice to avoid typing errors.

The password must be min. 8 characters and contain at least 3 types of the following requested types of characters:

- upper case letter
- lower case letter
- figure
- special character

💈 GP web	GP webpay Keystore Manager		
g	ECOMMERCE GP WEBPAY		
		Saving the private key file	
	Path directory	C:\klic	Browse
	New password	•••••	?
	Password confirmati	••••••	
Select the path directory for the private key and enter a new password. The password protects your key and it will be required for the operations within the Portal GP webpay.			
PRE	INIOUS	CA	ANCEL FINISH

As all necessary information is entered, it is possible to finish the action by pressing the "Finish" button. It is also possible to return to the previous step by pressing the "Previous" button, or to jump back to the start screen by pressing the "Cancel" button.

If a non-existing directory is entered, the following message is displayed, when you try to continue:

		e e e e e e e e e e e e e e e e e e e	3
ERRO	R The d	directory does not exist.	
		CLOSE	

In case of inequality of passwords, the following message is displayed:

		E
ERRC	Password does not match the confirmation password.	
	CLOSE	

If everything has been entered correctly, the key is converted and the following message is displayed:

SUCCESS	The private key format has been successfully updated.
	CLOSE

In the selected destination directory is created a file named "gpwebpay-pvk.key". The file contains the private key in the text format PEM.

By pressing the "Close" button, the application returns to the initial screen.

3.3.3 Password change

This option works with the new format of keystore and it is necessary to update the keystore file first – see the previous chapter – or to use the file in the new format obtained from the GP webpay Portal.

By pressing the "tile", a new window is opened to enter the necessary data:

💈 GP wet	opay Keystore Manager	
g	ECOMMERC GP WEBPAY	E
		Changing the private key password
	File path	Browse
	Password	Password
	New password	New password ?
	Password confir	Password confirmation
	Select th	ne private key file (with .key extension) and change password.
		CANCEL FINISH

First, it is necessary to find the directory containing the private key file by means of the "Browse" button:

Otevřít	
G G マ J → Počítač → OS (C:) → klic	✓ ♣ Prohledat: klic
Uspořádat 🔻 Nová složka	:≡ ▼ 🔟 🔞
ConeDrive Název položky	Datum změny Typ
G gpwebpay-pvk.key	7.3.2016 13:47 Soubor KI
 Dokumenty Hudba Obrázky Subversion Videa 	
🚢 OS (C:)	
🕳 Vyměnitelný disk 👻 🕢 🏢	•
<u>N</u> ázev souboru: gpwebpay-pvk.key	✓ PEM key files (*.pem;*.key) ✓ Otevřít ▼ Storno

and to "Open" the relevant file. It is also necessary to enter the password to the original key and the new password as well (and of course to verify the new password by entering it twice).

The password must be min. 8 characters and contain at least 3 types of the following requested types of characters:

- upper case letter
- lower case letter
- figure
- special character

If the incorrect old password is entered or if the file format is incorrect, the following message is displayed:

ERROR	Wrong password has been entered, or the expected content file is missing.
	CLOSE

If the new password is not entered or it does not meet the necessary security requirements, the following message is displayed:

ERROR	Entered password does not meet the length or the type of characters.	
	CLOSE	

If the new password does not match the confirmation password, this situation is indicated by the following message:

		2
ERROR	Password does not match the confirmation password.	
	CLOSE	

If the values are entered correctly, there is displayed the following confirmation of a successful password change:

SUCCESS	The password change has been successfully completed.
	CLOSE

And there follows a return to the start screen.

3.3.4 For developers

The section "FOR DEVELOPERS" is primarily devoted for programmers implementing the payment gateway into a merchant's e-shop.

Selection starts the process of converting the keystore format to the next most common keystore formats and simultaneously saves the public part of the key into the generally used formats.

Input format of the keystore:

• text format PEM (PVK) - gpwebpay-pvk.key

Output formats of keystores and files:

- keystore:
 - o JAVA format JCEKS gpwebpay-pvk.jceks
 - o Microsoft PKCS12 gpwebpay-pvk.p12
- public key file:
 - o text format PEM gpwebpay-pub.pem
 - o binary format DER gpwebpay-pub.cer

The first step of conversion is selection of the keystore file. Use the "Browse" button to select the keystore file:

	Otevřít	x
	🕞 🕞 🚽 🕨 Počítač 🕨 OS (C:) 🔸 klic 🔹 4-7 Prohledat: klic	9
	Uspořádat 🔻 Nová složka 🔠 🔻 🗍	0
GP webpay Keystore Manager	ConeDrive Název položky Datum změny Typ Velikost	
	gpwebpay-pvk.key 2.5.2016 15:50 Soubor KEY 2 kB	
GP WEBPAY	Dokumenty	
	Hudba	
Export the privat	Subversion	
	Videa	
File path	1 Počítač	
Password Password	🚢 OS (C.)	
	📼 OS ():) 👻	
From the selected private key (.key extension) the ap	Název souboru: gpwebpay-pvk.key PEM key files (*.pem;*.key)	-
JKS (JCEKS - Java KeyStore), con .p12 (PCKS#12 - Public Key Cryptographic St	Otevřít 🔻 Storno	
.pem (PEM - Privacy Enhanced Mail), cer (CEP - certificate file), cont	ins the certificate	
.cer (cert - certificate hie), conta	inis the certificate	
	CANCEL NEXT	

"Open" the file, enter password and press the "Next" button.

If the password or the inner keystore format is incorrect, the following message is displayed:



Otherwise, you are prompted for selection of output directory to save the new created files and to enter the new keystore password (the same password is used also to protect the private key in the keystore; the original password can be used as well).

The "Browse" button opens the window for selection of the output directory; after the necessary space in the file system is found, it is necessary to confirm the selection by the "Select folder" button:



then it is necessary to enter the password and confirm it.

The password must be min. 8 characters and contain at least 3 types of the following requested types of characters:

- upper case letter
- lower case letter
- figure

• special character

Then the "Finish" button is to be pressed.

If a new password is not entered, or if it does not meet the necessary security requirements, the following message is displayed:

ERROR	Entered password does not meet the length or the type of characters.
	CLOSE

If the new password is not identical to the password confirmation, the situation is indicated by the following message:

ERF	ROR	Password does not match the confirmation password.	
		CLOSE	

If all the values are entered correctly, the successful export of the private key is confirmed by the following message:

SUCCESS	The private key export has been successfully completed.	
	CLOSE	

This concludes the process of export of the private key, and in the selected output directory are created the following files:

@ @ - 	Počítač 🕨 OS (C:) 🕨 klic5	▶ pay		Prohledat: pay	ـــــــــــــــــــــــــــــــــــــ
Uspořádat 🔻	Zahrnout do knihovny 🔻	Sdílet s 🔻 Vypálit	Nová složka		• 🗌 🔞
👢 jre	*	Název položky	Datum změny	Тур	Velikost
L klic klic1 klic2 klic3 klic4	Ŧ	 gpwebpay-pub.cer gpwebpay-pub.pem gpwebpay-pvk.jceks gpwebpay-pvk.key gpwebpay-pvk.p12 	27.11.2019 10:22 27.11.2019 10:22 27.11.2019 10:22 27.11.2019 10:21 27.11.2019 10:21	Certifikát zabezpečení Soubor PEM Soubor JCEKS Soubor KEY Personal Information Exchange	1 kB 1 kB 3 kB 2 kB 3 kB
klic5	• Džky: 5				

- gpwebpay-pvk.key private key generated by GP webpay Portal
- gpwebpay-pvk.jceks private key in the keystore in JAVA language (JCEKS)
 - applicable for JSP/JAVA applications
- gpwebpay-pvk.p12 private key in the keystore in Microsoft structure (PKCS12 P12)
 - applicable for .NET applications
- gpwebpay-pub.pem text PEM (PVK) format of the public key

 applicable for PHP applications for verification of the accuracy of the signature value created by the private key

• gpwebpay-pub.cer - binary DER format of the public key

 applicable for .NET applications for verification of the accuracy of the signature value created by the private key

 format for sending the public key to the GP webpay application support, if recording of the public key by means of the GP webpay Portal fails

After pressing the "Close" button the application returns to the start screen.

4. Signing messages

4.1 General technical basis

4.1.1 Signing a request

(The complete example is in chapter Digest examples)

GP webpay accepts only requests for which it can be proved that the originator of the request is an authorized subject (i.e. merchant) with whom GPE, s.r.o. has signed a contract for GP webpay services.

The DIGEST field is used to prove the origin of the request. Its contents are generated based on the following data:

- Data sent this data is used to prove that the contents of the fields have not been changed on the way to the system.
- Private key the private key is used to prove that the request comes from the merchant.

At the moment of beginning the integration, the merchant using the GP webpay Portal generates a private key, which he/she stores securely and provides it to the developer for integration. The merchant's public key is stored automatically on the GP webpay server and before the merchant,s request is accepted, it will be used for verifying, if the merchant has signed the request by his/her private key.

DIGEST parameter contained in transmitted requests contains electronic signature of all other fields of the request. The electronic signature guarantees integrity and undeniableness of the transmitted request.

Any request not containing the DIGEST field or with non-matching contents of the DIGEST field will be rejected with the following explanation:

- PRCODE=5 SRCODE=34 "Mandatory field missing, DIGEST" or
- PRCODE =31 "Invalid signature".

To generate and verify the electronic signature, a string composed as a concatenation of the text interpretation of the values of all fields contained in the request sent, except from the DIGEST field. When compiling the input message, the merchant has to use the same order of fields as that used in the definition of the request and intersperse individual fields by delimiter "|" (pipe, ASCII 124, hexa 7C). The delimiter must not be preceded or followed by whitespace. <u>URLEncode parameters are used only for data transmission, original data have to be used to generate a signature</u>.

Source for generating the DIGEST field in case of method CREATE_ORDER is the value created by concatenation of the fields in the order given here:

MERCHANTNUMBER + | + OPERATION + | + ORDERNUMBER + | + AMOUNT + | + CURRENCY + | + DEPOSITFLAG + | + MERORDERNUM + | + URL + | + DESCRIPTION + | + MD

If the request does not contain any of optional fields, this field is skipped. If the field is sent empty, it is necessary to include it in generating for the DIGEST field and in the string, there will be two separators next to each other $- \parallel$.

If the merchant sends only obligatory parameters, for generating the DIGEST field serves the value: MERCHANTNUMBER + | + OPERATION + | + ORDERNUMBER + | + AMOUNT + | + CURRENCY + | + DEPOSITFLAG + | + URL

4.1.2 Response verification

All the responses from the GP webpay system also contain the DIGEST field. Its contents are generated as follows:

- based on the data contained in the response;
- and, at the same time based on the GP webpay private key.

At the moment of beginning the integration, the merchant downloads the GP webpay public key from the GP webpay Portal. It is used by the merchant to verify the contents of the DIGEST field.

This way the merchant can easily verify that:

- the response really comes from the GP webpay;
- the response has not been changed on the way to the merchant.

Furthermore, the response contains also the DIGEST1 parameter, which further enhances the security of the response. The DIGEST1 parameter is generated as the DIGEST parameter, but parameter "MERCHANTNUMBER" is added to the parameters for validation of the DIGEST parameter. This parameter is not sent in the response and the merchant has to add it by himself/herself because he/she knows its value.

The resulting string for validation of the DIGEST1 field looks like this:

<string for the field DIGEST> + | + MERCHANTNUMBER

4.1.3 Generating the electronic signature

Inputs:

- Data message (message)
- Private RSA key (with a K-length modulus)

Outputs:

• Electronic signature (BASE64 encoded), approximate length K*1.5

The electronic signature is generated as follows

a) the value of the function SHA-1 [3] is derived from the message

 b) the hash is encoded into the input value for the RSA signature, using the EMSA-PKCS1v1_5-ENCODE algorithm as described in paragraph 9.2.1 [1]. The encoding is made as follows:

01 | FF* | 00 | 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14 | hash

where FF characters are repeated as many times as necessary for the total length of the string to be one octet shorter than the key modulus. The character | is used for the strings concatenation.

- c) the RSA signature is calculated using the output value from b), as described in 8.1.1 [1] RSASSA-PKCS1-V1_5-SIGN
- d) The output from c) is encoded using BASE64

4.1.4 Verification of the electronic signature

Inputs:

- Data message
- Electronic signature (BASE64 encoded)
- Public RSA key

Outputs:

- Logical value YES the signature is valid
- Logical value NO the signature is invalid or its verification has not been possible.

The electronic signature is verified as described in 8.1.2 [1] in the following main steps:

- a) depending on the settings for the merchant in the GPE system, the correct public key is selected and its integrity is verified;
- b) the electronic signature is decoded using BASE64;
- c) the output from b) is decrypted using the selected public key;
- d) a miniature (hash) is generated based on the message and encoded as described in "Generating of the electronic signature", paras a) and b);
- e) the electronic signature decoded according to c) is compared with the result from d). If they are identical, the function returns a logical truth (the signature is valid).

Otherwise, the function returns a logical untruth (the signature is not valid).

The application used for verification of the electronic signature has to identify a signature as invalid also in the case, if verification of the signature has not been possible (for example, due to unavailability of the key).

4.1.5 Graphic representation of key generation and verification



4.1.6 Keys used

To generate the electronic signature (DIGEST), RSA keys (keyPair) are used with a modulus length of 2048 bits. During the communication between GP webpay and the merchant, the following key pairs are used:

	GPE's private key (GPE _{PRI})	Used for the calculation of the electronic signature for messages sent by GPE.	
GPE's KeyPair	GPE's public key (certificate) (GPE _{PUB})	Used by the merchant to verify the electronic signature in messages sent by GPE.	Delivered in the form of a X509 certificate

Merchant's	Merchant's private key (MERCH _{PRI})	Used for generating the electronic signature for messages sent by the merchant.	
KeyPair	Merchant's public key (certificate) (MERCH _{PUB})	Used by GPE to verify the electronic signature in messages sent by the merchant.	Delivered in the form of a X509 self-signed certificate

The application used to generate a self-signed certificate is delivered to the merchant when the merchant applies GPE, s.r.o. for signing a contract. Commercially issued keys can be used as well, but their validity is limited to 1 or 2 years (in comparison with the key generated by the application, there the key validity is longer).

4.1.7 Logging

The application used to verify the electronic signature must store in its audit logs all information about successful and non-successful verification of the electronic signature.

For the purpose of verification of the audit logs, all data required for the verification and reverification of the electronic signature must be logged. This data includes mainly the electronic signature, the fields, which have been used for its generation, and the result of its verification. If any logs are missing or incomplete, the authenticity of such transactions cannot be confirmed.

4.1.8 References

For further information about the mechanism used to generate the DIGEST field, see the following documents:

- [1] RFC 2437, PKCS #1: RSA Cryptography Specifications, October 1998;
- [2] XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002,

http://www.w3.org/TR/xmldsig-core/;

- [3] RFC 3174 US Secure Hash Algorithm 1 (SHA1), September 2001;
- [4] RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile,

January 1999

The following cryptographic libraries and components may be used to generate the electronic signature:

- JCE Cryptix: Alternative JCE provider offering an algorithm for the RSA/SHA1/PKCS#1 signature, <u>www.cryptix.org</u>
- Bouncy Castle: Alternative JCA provider offering libraries for the generation of certificates and work with the PKCS#12 certificate storage, www.bouncycastle.org.

Crypto++, a free C++ class library of cryptographic schemes supporting also the RSA/SHA1/PKCS#1 algorithm, <u>www.cryptopp.com</u>

4.2 Digest examples

4.2.1 Test key and application ZIP file

Due to security reasons, it is not possible to insert application into document and send it via e-mail. You have to download it from GP webpay Portal from "Download section":

Dash	nboa 🗙 🛛 🗖 MP	2I - Da 🗙 👸 Net	cete: 🗙 📋 502 Pro: 🗙 🛛 🛃 Seznam 🗙 🛛 👸	localhos X 😢	Global P × +	
$\leftarrow \ \rightarrow$	C 🔒 http	s://test.portal.gpw	ebpay.com/portal/index.xhtml	5	🗠 🝈 🛛 🚸 💆	🔩 🗐 E
Aplika	ace 💈 GP webp	oay - Demo 👳 I	Mobizen Your An Srovnání stavebníc	Netcetera 📙 P	řehled zpráv 👖 TV 📙	Moje »
E GF	COMME P WEBPAY	RCE		Ŕ	AA Admin A	Admin V MUZO
	1711000	Documentation	or reopay intrate key management	01 0411 2017		
$\overline{\bigcirc}$	PAY006	Documentation	GP webpay API HTTP	17-Jan-2019		
	PAY006	Documentation	GP webpay API WS	15-Nov-2018		
A	PAY006	Documentation	GPE test public key ("GPE_test_public_key.zip" file)	31-Jan-2017	Download	
	PAY006	Documentation	Logos – GP webpay and payment methods	02-Mar-2017	Download	
	PAY006	JNLP	Application – GP webpay Keystore Manager	12-Jul-2017	Download	
$\hat{\mathbf{O}}$	PAY006	Documentation	Application – Demo e-shop PHP	08-Mar-2017	Download	- 18
	PAY006	Documentation	GP webpay - transfer from GUI to Portal GP Webpay	03-Mar-2017		
8	PAY006	Documentation	GP Webpay - Transition to the TLS 1.2 protocol	29-Jan-2018		1 💌 👘
R	PAY006	Documentation	GP webpay - WS API - WSDL	15-Nov-2018	Download	
Ψ	PAY006	Documentation	GP webpay - implementation examples	24-May-2017	Download	
් උ	XLSX 🗴 PDF	ß	1 - 13 / 13			20 🔻
© Global Payme	ents Europe 2016, v: 1.12.0	0			English	•

From downloaded zip file copy this files into directory unzipped from "digest_app.zip" file.

```
Directory in downloaded ZIP - "overovani_podpisu-digest_verification", files
```

"run.cmd", "digestProc.exe", "digestProc.jar".

The final directory should be:

					* •
↑ Název	Přípo	Velikost	Datum		Atrib
€ []		<dir></dir>	26.04.2019	13:24	_
Contract Con	exe	38 713	25.11.2014	09:28	-a
🛓 digest Proc	jar	20 793	25.11.2014	09:28	-a
Gigest Proc	proper	ties 532	26.04.2019	11:54	-a
gpe.signing_prod	cer	794	15.08.2013	09:57	-a
🔄 gpe.signing_test	сег	804	15.08.2013	09:57	-a
🔄 gpwebpay-pub-test	сег	868	26.04.2019	11:14	-a
🗞 gpwebpay-pvk-test	jks	2 220	26.04.2019	11:55	-a
🚳 run	cmd	33	13.10.2015	15:22	-a

Files

digestProc.exe - the application executable in the MS Windows environment

digestProc.jar - Java archive of the application

gpwebpay-pub-test.jks-testing keystore

gpwebpay-pub-test.cer - testing public key (certificate)

digestProc.properties - configuration file

gpe.signing prod.cer - GPE production environment public key

gpe.signing test.cer - GPE testing environment public key

Configuration file

Content of the file	Meaning in English
####### Privatni klic #######	####### Private key #######
#nazev keystore s privatnim klicem	#name of the keystore containing the
	private key
keyStoreFile = gpwebpay-pvk-test.jks	keyStoreFile = gpwebpay-pvk-test.jks
#nesio ke keystore	#password to the Keystore
Himono (alias) privatniho klico	the store was a store the private key
mjmeno (allas) plivatnino kilce privateKevAlias = alias	miane (allas) of the private key privateKevAlias = alias
#heslo k privatnimu klici	#password to the private key
privateKevPwd = Abcd1234	privateKeyPwd = Abcd1234
####### Privatni klic #######	####### Private key #######
	ak
####### Verejny klic #######	####### Public key #######
#jmeno souboru s verejnym klicem – cizi	#name of the file containing the public key
verejny klic	- somebody else's public key
<pre>publicKeyFile = gpwebpay-pub-test.cer</pre>	<pre>publicKeyFile = gpwebpay-pub-test.cer</pre>
<pre>#publicKeyFile = gpe.signing_test.cer</pre>	<pre>#publicKeyFile = gpe.signing_test.cer</pre>
<pre>#publickeyFile = gpe.signing_prod.cer ######## Voroiny klic ########</pre>	<pre>#publickeyFile = gpe.signing_prod.cer ######## Public kow ########</pre>
####### Verejny Kiic #######	####### PUDIIC Key ########
####### URLEncoding #######	####### URLEncoding #######
#predepsane enkodovani vstupnich parametru	#required encoding of input parameters
encoding = UTF-8	encoding = UTF-8
#digest na vstupu je URLEncoded	
<pre># input_urlencoded = true</pre>	#input digest is URLEncoded
input_urlencoded = false	<pre># input_urlencoded = true</pre>
	input_urlencoded = false
#na vyslednem digestu udelat URLEncode	#to make UPIEncode over the output direct
# output urlencoded = true	# output urlencoded = true
autput urlencoded = false	<pre>autput_urlencoded = false</pre>
####### URLEncoding #######	####### URLEncoding #######
······································	······································

Start-up

Java applications need for their run the so called Java Runtime. Java Runtime is a runtime environment and it is necessary to be installed additionally into the majority of operation systems. Java Runtime is freely downloadable from:

http://www.java.com

As concerns the offered versions, the Java SE JRE (Standard Edition, Java Runtime Environment) version is absolutely sufficient. The application is functional also with Java Runtime of other providers.

If you start up the application without parameters:

digestProc.exe

java -jar digestProc.jar

then help is displayed:

Help displayed:	Meaning in English
Digest Processing, (c) 08/2004 Dimitrij R. Holovka	Digest Processing, (c) 08/2004 Dimitrij R. Holovka
<pre>Pouziti: - vypocet podpisu: java -jar digestProc.jar -s <retezec pro="" vypocet=""> - overeni podpisu: java -jar digestProc.jar -v <retezec pro="" vypocet=""> <podpis> nebo</podpis></retezec></retezec></pre>	Usage: - generation of signature: java -jar digestProc.jar -s <string for="" generation=""> - verification of signature: java -jar digestProc.jar -v < string for generation> <digest></digest></string>
1000	or
<pre>- vypocet podpisu: digestProc.exe -s <retezec pro="" vypocet=""> - overeni podpisu: digestProc.exe -v</retezec></pre>	 generation of signature: digestProc.exe - s <string for="" generation=""></string> verification of signature: digestProc.exe
<retezec pro="" vypocet=""> <podpis></podpis></retezec>	-v <string for="" generation=""> <digest></digest></string>
Pokud <retezec pro="" vypocet=""> obsahuje mezery je nutne jej dat do uvozovek.</retezec>	If <string for="" generation=""> contains spaces, it is necessary to put it into quotation marks.</string>

Examples:

Generation of signature:

Start-up: digestProc.exe -s "hello world"

Result:

Result displayed:	Meaning in English
Digest Processing, (c) 08/2004 Dimitrij R.	Digest Processing, (c) 08/2004 Dimitrij R.
HOLOVKA	HOLOVKA
Soubor keyStore: test ks	File keyStore: test ks
Heslo keyStore: changeit	KeyStore password: changeit
Alias privatniho klice: paymuzo	Alias of the private key: paymuzo
Heslo privatniho klice: changeit	Private key password: changeit
Privatni klic: OK	Private key: OK
Verejny klic: OK	Public key: OK
Soubor publicKeyStore: test.cer Verejny klic: OK	File publicKeyStore: test.cer Public key: OK
URLEncoded podpis na vstupu: false Provadet URLEncode na vystupu: false Kodova stranka: WINDOWS-1250	Input URLEncoded signature: false To make URLEncode on the output: false Code page: WINDOWS-1250
Vytvoreni podpisu pomoci privatniho klice: Delka podpisu: 344 Soubor s podpisem: digestProc.sign Podpis:	Generation of signature using the private key: Length of signature: 344 File with signature: digestProc.sign Signature:

Signature:

et2dmSt7+9y43r6pCjAAiLKzDws9VgAm/Qh1ZZxyy1QWS43WLH1ywUHBS2f68hXPARAMPOaCpLiswz8iS3pameakq wTdlmeff6hOAR2s1/ACToo/dICYZoCkXdXg8dKzHqcWVnqeiGBY+NmewyH2/KHg/IzxxqF6AeLbTJsH5drPlUBVsT bRprUfNAzoCizLRLM3fLXvy8bvDte35cq7Syly6snFjo3crUBwEzXLLrjU/XEYPvzL/XwNo3IZ7PhGspdx8gSWYj7 7t2zasLh/AxpkI8qYqwQlEYzmBHKB3NFetjJoR9jDCvFFtxMvI4bHYHKnjJ1mPyBPxJp00sxp/w==

The application generates – by means of the **private key** "gpwebpay" stored in keystore "test.ks" - signature of the string "hello world". The signature is displayed on the screen and at the same time is saved in the "digestProc.sign" file.

Verification of the signature:

Start-up:

```
digestProc.exe -v "hello world"
```

Result:

Result displayed:	Meaning in English
Digest Processing, (c) 08/2004 Dimitrij R.	Digest Processing, (c) 08/2004 Dimitrij R.
Holovka	Holovka
Nacitani parametru OK	Reading of parameters OK
Soubor keyStore: test.ks	KeyStore file: test.ks
Heslo keyStore: changeit	KeyStore password: changeit
Alias privatniho klice: paymuzo	Alias of the private key: paymuzo
Heslo privatniho klice: changeit	Private key password: changeit
Privatni klic: OK	Private key: OK
Verejny klic: OK	Public key: OK
Soubor publicKeyStore: test.cer	File publicKeyStore: test.cer
Verejny klic: OK	Public key: OK
URLEncoded podpis na vstupu: false	Input URLEncoded signature: false
Provadet URLEncode na vystupu: false	To make URLEncode on the output: false
Kodova stranka: WINDOWS-1250	Code page: WINDOWS-1250
Overeni podpisu pomoci verejneho klice: Vysledek overeni: true	Verification of signature using the public key: Result of verification: true

The application verifies the string signature "hello world" by means of the public key (certificate) stored in the "test.cer" file. The result of verification is displayed on the last line – "Vysledek

overeni: true" (i.e. "Verification Result: true").

To verify the response of the GP webpay systems it is necessary to change the parameter "publicKeyFile" in the configuration file as follows:

- Testing environment: gpe.signing_test.cer
- Production environment: gpe.signing prod.cer

4.2.2 Digest example

! This is only example, it doesn't work in real testing environment (it uses the test key without corresponding public part on GP webpay server) !

If an "Invalid keystore format" error occurs when trying to sign/verify a string, the different type of keystore (JCEKS, P12...) than the supported "JKS" is used:



You need to convert the used private keystore to the correct format using one of the key-handling programs – e.g. KeyStore Explorer, OpenSSL...

4.2.2.1 HTTP

4.2.2.1.1 Request

HTTP request

https://test.3dsecure.gpwebpay.com/pgw/order.do?MERCHANTNUMBER=9999999021&OPERATION=CREATE_ORDER&ORD ERNUMBER=157487125803&AMOUNT=100&CURRENCY=203&DEPOSITFLAG=1&MERORDERNUM=155912254545&URL=https%3A%2F %2Flocalhost%3A443%2Fdemoshop%2Fpayment%2Fpayment.php&userparam2=59452C6A0381B48B3B164A80E202983F542 759CC17AF36DE37B4CDB4B9908EB7&email=dholovka%40gpe.cz&DIGEST=BZ1XwJJGku1m2uhyHU%2ByIyhzRv%2BVKfNd1%2 F1Y5pSG61FfJb88JzY1Ls7CGfhY6%2Bonle3q0sDH6V%2FM9EF4uIWMcSJxBpqzPmh7U6ud7nYa3UINr71jSU3WC8FQd8qIc%2FL WeVFgh%2BPgxq0cdI47vvFZXu0BtzFpZQZWI0AOOQxhKAzZ4UrRafdNGYKtYI1BwZD4u5Aq3wSr00BRBu%2BFompn%2BIWAN8xpA ySA2A9VvBEHGvD1tFLSMIV1CCOMfqq0EEQj0TceXrWoQ8Y02gQoo1RcCDb1geOsshELdaseuKUtcFnsZE49%2B700G11xYz9%2F4 RNMOTa%2FPpwTVEc00qlqlgfjWQ%3D%3D

Data for digest:

9999999021|CREATE_ORDER|157487125803|100|203|1|155912254545|https: //localhost:443/demoshop/payment/payment.php|59452C6A0381B48B3B164 A80E202983F542759CC17AF36DE37B4CDB4B9908EB7|dholovka@gpe.cz

Digest:

BZlXwJJGkulm2uhyHU+yIyhzRv+VKfNdI/1Y5pSG61FfJb88JzYlLs7CGfhY6+onle 3q0sDH6V/M9EF4uIWMcSJxBpqzPmh7U6ud7nYa3UINr71jSU3WC8FQd8qIc/LWeVFg h+Pgxq0cdI47vvFZXuoBtzFpZQZWI0AOOQxhKAzZ4UrRafdNGYKtYI1BwZD4u5Aq3w SrOOBRBu+Fompn+IWAN8xpAySA2A9VvBEHGvD1tFLSMIV1CCOMfqq0EEQj0TceXrWo Q8Y02gQoolRcCDb1geOsshELdaseuKUtcFnsZE49+700G11xYz9/4RNMOTa/PpwTVB c00qlqlgfjWQ== Command to start the application:

```
java -jar digestProc.jar -s
"9999999021|CREATE_ORDER|157487125803|100|203|1|155912254545|https
://localhost:443/demoshop/payment/payment.php|59452C6A0381B48B3B16
4A80E202983F542759CC17AF36DE37B4CDB4B9908EB7|dholovka@gpe.cz"
```

The result is displayed on the screen and saved in the file "digestProc.sign":

Správce: C:\Windows\SysWOW64\cmd.exe	Carlo	
C:\Digest>run -s "999999021 CREATE_ORDER 157487125803 100 203 1 155912254545 https://locall &&DE202983F542759CC17AF36DE37B4CDB409908E87 dholovka@gpe.cz"	host:443/demoshop/payment/payment.php	59452C6AD381B48B3B164
C:\Digest>java _jar_digestProc.jar_s_"999999021[CREATE_ORDER]157487125803[100]203[1]155912 594526680381848838164880E202983F542759C617AF360E37D4C08489908EB7[dholovka@gpe.cz"	2254545¦https://localhost:443/demosho	p/payment/payment.php;
Digest Processing, (c) 10/2014 Dinitrij R. Holovka		
Nacitani parametru OK Soubor keyStore: gpwebpay-pvk.jks Heslo keyStore: Abed1234 Alias privatniho klice: alias Heslo privatniho klice: Abed1234 Privatni klic: OK Soubor publicKeyStore: gpe.signing_test.cer Verejny klic: OK		
URLEncoded podpis na vstupu: false Provadet URLEncode na vystupu: false Kodova stranka: UTF-8		
Uytvoreni podpisu pomoci privatniho klice: Delka podpisu: 344 Soubor s podpisen: digestProc.sign Data k podpisu: "999999901 EREATE_ORDER!157487125803;100!203;11155912254545;https://localho BUE20298155425950f1747360E3948CB0689908EB7]dholovka@gpe.cz" Podpis: PilkuJdsku1m2uhyHU+yTyhzRv+UKfHdT/1Y5p5661FfJb80JzYILs7C6fhY6+onle3q05DH6U/M9EF&uTWMcSJxBpq oBtzFp2Q2VIDA00UxhKdzZ4UrRafdH6VKtY11Bw2D4u5dq3wSr00BRBu+Fompn+TWAN6xpAySA2A9UvBEH6vB1tFLSM Zf4974D6f1XY29/ARMH0Ta/PpuTUBeD101a11gf1W0=	ost:443/demoshop/payment/payment.php 2Pmh7U6ud7nYa3UINr71j\$U3WC8FQd8qIc/LW IVICC0MfqqDEEQj0TceXrWoQ8Y02gQooIncCU	59452C6A0381B48B3B164A eVFgh+Pgxq0cd147uvFZXu b1ge0sshELdaseuKUtcFns
C:\Digest>		T

4.2.2.1.2 Response

HTTP response

https://localhost:443/demoshop/payment/payment.php?OPERATION=CREATE_ORDER&ORDERNUMBER=157487125803&M ERORDERNUM=155912254545&PRCODE=0&SRCODE=0&RESULTTEXT=OK&DETAILS=59452c6a0381b48b3b164a80e202983f8e9c 5459c948e292465bc638b8be647d&USERPARAM1=2F89879EAF57B52B37E23DFD1D2B1BA6567A13BC2547F16DBB54EF5BE3A7 43A7&TOKEN=AA74E7D735D3201A926971BE5A92C8CE14D2E685DC399E4A3E2BE12C64605EC7&EXPIRY=2012&ACCODE=69Z41 V&ACSRES=A&PANPATTERN=405607*****0016&DAYTOCAPTURE=04122019&ACRC=00&RRN=000001267633&DIGEST=o9uwRov 2%2BWkF5Qc25EhL0Ka7d8c0bEW2n59j62WFDWcAL5HUD2g5%2F91wfo3ZD2EKvaNtXmn9QYLXK4mwlgRWpdjCtuYgJxmHBa%2BSE %2FfUSRVBV3y2Qt47eLZNH8UGtSX1Hy3IzGt%2FCX4UYe0woZ%2BI4keiQr1F%2FRC7w6A0QCLM5rSEtNhM487eM8A5k12E%2Bpu zQeVtSr9X3nCa3N531N1Fm9p96bQKejpI7nX9%2FJJk8pk6n0MXBGzYbxX%2B0YNt2sUlCfFJrCg44LWQsBZY11vL1xHdZHfKK6F 6LBwMTAb0qD5zE1fQUdzLYPWFalb4tTtHC6srwv3p1e5A0Z3efWzPOA%3D%3D&DIGEST1=hudnVKmuIHpBG4U4BXsNGTdxoi0q98 60K%2Fntgher8gwHab2TOavI5DCNs2hURSAC9nX4nDJa30g2E7FVigWI%2BYf45kHgQ2MQEASGQg11TDqyHRoYFZwqM69hXH0Ugd 3RXEZDmYnLm6VvJ4PvVKqCa8FBC2JUyo9saKN2rXVjv2yox265u724r7JoIIGg5ka%2F5Schu7bxRuG%2BdID4YNePR4gz9C6K9V 8b2bWQ%2FifY0EHGtySMfZ7BrPIDXRNRJM5YrnGUlLghsYfpJwI%2Bp4d5kpUSHkiwQ5yNXhLPI8xER%2BC0%2Bmve9xV6n%2Fvp hiQo8T1RBcp2bAK4IOAr5%2FT7%2BTBVtQ%3D83D

Data for verification:

CREATE_ORDER|157487125803|155912254545|0|0|0K|59452c6a0381b48b3b16 4a80e202983f8e9c5459c948e292465bc638b8be647d|2F89879EAF57B52B37E23 DFD1D2B1BA6567A13BC2547F16DBB54EF5BE3A743A7|AA74E7D735D3201A926971 BE5A92C8CE14D2E685DC399E4A3E2BE12C64605EC7|2012|A|69Z4IV|405607*** ***0016|04122019|00|000001267633|9999999021

Digest (DIGEST1 – URLDecoded):

hudnVKmuIHpBG4U4BXsNGTdxoiOq9860K/ntgher8gwHab2TOavI5DCNs2hURSAC9n X4nDJa30g2E7FVigWI+Yf45kHgQ2MQEASGQg11TDqyHRoYFZwqM69hXHoUgd3RXEZD mYnLm6VvJ4PvVKqCa8FBC2JUyo9saKN2rXVjv2yox265u724r7JoI1Gg5ka/5Schu7 bxRuG+dID4YNePR4gz9C6K9V8b2bWQ/ifYOEHGtySMfZ7BrPIDXRNRJM5YrnGUlLgh sYfpJwI+p4d5kpUSHkiwQ5yNXhLPI8xER+CO+mve9xV6n/vphiQo8T1RBcp2bAK4I0 Ar5/T7+TBVtQ==

Command to start the application:

java -jar digestProc.jar -v

"CREATE_ORDER|157487125803|155912254545|0|0|0K|59452c6a0381b48b3b1 64a80e202983f8e9c5459c948e292465bc638b8be647d|2F89879EAF57B52B37E2 3DFD1D2B1BA6567A13BC2547F16DBB54EF5BE3A743A7|AA74E7D735D3201A92697 1BE5A92C8CE14D2E685DC399E4A3E2BE12C64605EC7|2012|A|69Z4IV|405607** ****0016|04122019|00|000001267633|9999999021"

"hudnVKmuIHpBG4U4BXsNGTdxoiOq9860K/ntgher8gwHab2TOavI5DCNs2hURSAC9 nX4nDJa30g2E7FVigWI+Yf45kHgQ2MQEASGQg11TDqyHRoYFZwqM69hXHoUgd3RXEZ DmYnLm6VvJ4PvVKqCa8FBC2JUyo9saKN2rXVjv2yox265u724r7JoIlGg5ka/5Schu 7bxRuG+dID4YNePR4gz9C6K9V8b2bWQ/ifYOEHGtySMfZ7BrPIDXRNRJM5YrnGUlLg hsYfpJwI+p4d5kpUSHkiwQ5yNXhLPI8xER+CO+mve9xV6n/vphiQo8T1RBcp2bAK4I OAr5/T7+TBVtQ=="

The result is displayed on the screen:

🖬 Správce: C:\Windows\SysWOW64\cmd.exe	J
C:\Digest>run -v "CREATE_ORDER 157487125803 155912254545 0 0 0K 59452c6a0381b48b3b164a80e202983f8e9c5459c948c292465bc638b8be647d 2F89879EAF57B52837E23 DFD102818a6567a138c2547F16D8B54EF5BE3a743a7 aa74E7073503201a926971BE5a92C8CE1402E685Dc399E4a3E2BE12C64605EC7 2012 a 69741U 405607******0016 04122019 0 D 00001267633}9999992121" "hudnuKmu HpB64U4BX5NGTdxo10q986UK/ntgher3gwHab210av15DCNs2hURSNC9nX4nDJa30g2E7FUigU1+YF45KHq02MQEA56G9117DqyHRoYFZwqM65hXH OUgd3REZEDArbLn660J4PVUR6ca8F8C2JUu9938AK7R2ND2V02765U72477J3116g5ka/S5cH7U2N&RG+d1D4YNePR4g29C6K9U8b2bVQ/ifY0EH6tySMF3DFPTDXRNGJMSYrnGU1LghsYFpJw1+ p4d5kpUSHkiwQSyNXhLP18xER+C0+mve9xU6n/vphiQo8T1RBcp2bAK410Ar5/T7+T8UtQ=="	
C:\Digest>java _jar digestProc.jar _v "CREATE_ORDER 157487125803 155912254545 0 0 0R 59452c6a0381b48b3b164a80e202983f8e9c5459c948e292465bc638b8be647d 2768979EnE57852837E230FD102b18h6567A13BC2547F16DBB54EF5BE3874387 ah74E7D735D3201h926971BE5A92E8E14D2E685DC399E4A3E2BE12C646D5EC7 2012 1h 692AU 4D56D7 ******0616[04122019]00 000D010E7653]999999021" "budnukmu HpB6404BX:%bT4xo104986BX:btgher3ptab2Tav15DcH25AB7EA3E2BE12C646D5EC7 2012 1h 692AU 4D56D7 g]1TDayHBoYF2uqM69hXHoUga3BXE2DmVnLm6UvJ4PvUKqCa8FB62JUyo9saKH2rXUjv2yox265u724r7Jv116g5Ka/5Schu7bxRuG+d1D4YNePR4gz9C6K9U8b2bVQ/ifY0EH6tySMF27BrP1DXRN RJMSYrnGU1LghsYFpJvI+p4d5kpUSHkiwQ5yHXhLP1BxER+C0+mve9xV6n/vphiQo8T1RBcp2bAK4IOAr5/T7+TBUtQ=="	
Digest Processing, (c) 10/2014 Dimitrij R. Holovka Hacitani parametru OK Soubor keyStore: gpuebpay-pok.jks Heslo keyStore: Abc41234 Blias privatniho klice: alias Heslo privatniho klice: Abc41234 Privatni klic: OK Soubor publicKeyStore: gpe.signing_test.cer Uerejny klic: OK	
UBLEncoded podpis na vstupu: false Provadet UBLEncode na vystupu: false Kodova stranka: UTF-8	ł
Overeni podpisu pomoci verejneho klice: Vysledek overeni: true	
C:\Digest>	

4.2.2.2 WS

4.2.2.2.1 Request

WS request
<soapenv:envelope <="" td="" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"></soapenv:envelope>
<pre>xmlns:v1="http://gpe.cz/pay/pay-ws/proc/v1" xmlns:type="http://gpe.cz/pay/pay-ws/proc/v1/type"></pre>
<soapenv:header></soapenv:header>
<soapenv:body></soapenv:body>
<vl:getpaymentstatus></vl:getpaymentstatus>
<v1:paymentstatusrequest></v1:paymentstatusrequest>
<type:messageid>20191127174308776</type:messageid>
<type:provider>0100</type:provider>
<type:merchantnumber>9999999021</type:merchantnumber>
<type:paymentnumber>1</type:paymentnumber>
<type:signature></type:signature>
$\tt lTFcQSdDX9BMwup7B3YgC9SWKUROQYbuIe4TZmtGIFEP4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPsblaserstep4kAMsLkkHMmFCJjJcDtaU3x5NVTVSJTSel2J1Q8DIemhf/2eWWmFCJjNthf/2eWWmfCJftheq4kAMsLkkththfftheq4kAMsLkkththftheq4kAMsLkkththftheq4kAMsLkkthththftheq4kAMsLkkththththththththththththththththththt$
Dj/E/bg2mZQINr0R8c2OAMnXyffEX1ky3s1siAcu2aBDIYgpB9wi9fVvhmpIkLQY2QzDCrZtPMzsvLZ54v4uOScgXgTLwXxSeS04
x DZQ 1A8 Ng 6 o j X/d + 1 I v d PEk 4 q D 9 R j D K f u M v R j 1 W O 4 i + c Q PY 9 u 9 Q N 3 a J t m da 2 V O 3 J i c o M b V h U E T 1 H d j H D / x A c A R S 6 S t 8 / x I f u z g 2 5 S h a c
<pre>SpoOrQnjCwB/dNNPRsx5F+NUrZvDY8uPxqjCGuwayA==</pre>

Data for digest: 20191127174308776|0100|9999999021|1

Digest:

lTFcQSdDX9BMwup7B3YgC9SWKUROQYbuIe4TZmtGIFEP4kAMsLkkHMmFCJjJcDtaU3 x5NVTVSJTSel2J1Q8DIemhf/2eWWmDgPpsDj/E/bg2mZQINr0R8c2OAMnXyffEX1ky 3s1siAcu2aBDIYgpB9wi9fVvhmpIkLQY2QzDCrZtPMzsvLZ54v4uOScgXgTLwXxSeS 04xDZQ1A8Ng6ojX/d+1IvdPEk4qD9RjDKfuMvRj1WO4i+cQPY9u9QN3aJtmda2VO3J icoMbVhUET1HdjHD/xAcARS6St8/xIfuzg25SpoOrQnjCwB/dNNPRsx5F+NUrZvDY8 uPxqjCGuwayA==

Command to start the application:

java -jar digestProc.jar -s "20191127174308776|0100|9999999021|1"

The result is displayed on the screen and saved in the file "digestProc.sign":



4.2.2.2.2 Response

WS response
<soapenv:envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"></soapenv:envelope>
<soapenv:body></soapenv:body>
<ns4:getpaymentstatusresponse <="" td="" xmlns:ns4="http://gpe.cz/pay/pay-ws/proc/v1"></ns4:getpaymentstatusresponse>
xmlns="http://gpe.cz/gpwebpay/additionalInfo/response"
xmlns:ns5="http://gpe.cz/gpwebpay/additionalInfo/response/v1" xmlns:ns2="http://gpe.cz/pay/pay-
ws/core/type" xmlns:ns3="http://gpe.cz/pay/pay-ws/proc/v1/type">
<ns4:paymentstatusresponse></ns4:paymentstatusresponse>
<ns3:messageid>20191127174308776</ns3:messageid>
<ns3:state>1</ns3:state>
<ns3:status>UNPAID</ns3:status>
<ns3:substatus>INITIATED</ns3:substatus>
<pre><ns3:signature>M49XARQA7zu9xPHzaTgVUbFjpgXSC700H+BiN46mVldTrfqJYoSoQ9yAmcVrwF9Czv82ubHpY5+Q14MeQLMc0</ns3:signature></pre>
iK7vohR0jZscHHnMDUrpqj3l5WAaU/LEN/c0SR6dJDqHlbD1wvW557dTrTh3yjZWHT/bAqCqNU5S9rGVGNjB4kPcnxEPUsrApBuM
Gb+/Nugyf9VMUTdWeEbfuvgYfa//7fRUwABa5bhryiJW+1dvslR9MFaMYgqfhLeGDr+q8SaRIfE4qd/4XGKKF8Aa1x1hWldPVCKV
<pre>4pfag0/RcimaQ7IBb2IYV2rdtYyKhRwPj0WSU4OnkDzkU6MnauD3iRySA==</pre>

Data for digest: 20191127174308776|1|UNPAID|INITIATED

Digest:

M49XARQA7zu9xPHzaTgVUbFjpgXSC700H+BiN46mVldTrfqJYoSoQ9yAmcVrwF9Czv 82ubHpY5+Q14MeQLMc0iK7vohROjZscHHnMDUrpqj315WAaU/LEN/c0SR6dJDqHlbD 1wvW557dTrTh3yjZWHT/bAqCqNU5S9rGVGNjB4kPcnxEPUsrApBuMGb+/Nugyf9VMU TdWeEbfuvgYfa//7fRUwABa5bhryiJW+1dvs1R9MFaMYgqfhLeGDr+q8SaRIfE4qd/ 4XGKKF8Aa1x1hWldPVCKV4pfag0/RcimaQ7IBb2IYV2rdtYyKhRwPj0WSU40nkDzkU 6MnauD3iRySA==

Command to start the application:

java -jar digestProc.jar -v "20191127174308776|1|UNPAID|INITIATED" "M49XARQA7zu9xPHzaTgVUbFjpgXSC70OH+BiN46mVldTrfqJYoSoQ9yAmcVrwF9Cz v82ubHpY5+Q14MeQLMc0iK7vohROjZscHHnMDUrpqj315WAaU/LEN/c0SR6dJDqHlb D1wvW557dTrTh3yjZWHT/bAqCqNU5S9rGVGNjB4kPcnxEPUsrApBuMGb+/Nugyf9VM UTdWeEbfuvgYfa//7fRUwABa5bhryiJW+1dvs1R9MFaMYgqfhLeGDr+q8SaRIfE4qd /4XGKKF8Aa1xlhWldPVCKV4pfag0/RcimaQ7IBb2IYV2rdtYyKhRwPj0WSU40nkDzk U6MnauD3iRySA=="

The result is displayed on the screen:

