



Payment Gateway

IGFS[®]

opzione **BUY NOW**

**SPECIFICHE TECNICHE:
WEB SERVICES**

Ed. 09 del 24/11/2014



Networks & Transactional Systems

PREFAZIONE

SCOPO DEL DOCUMENTO

Il presente documento si propone di illustrare l'utilizzo dei Web Services o delle corrispondenti API realizzate da N&TS, per l'integrazione con sistema di pagamento elettronico IGFS® opzione **BUY NOW** di N&TS.

A CHI È RIVOLTO

Il documento è rivolto ai programmatori che sviluppano applicazioni Web destinate all'integrazione con sistemi di pagamento elettronico.

IGFS® opzione BUY NOW						
RELEASE PROD.	DOCUMENTO	ED. DOC.	DATA DOC.	STATO DOC.	AUTORE / I	VERIFICATA PROC. DOC.
2.1	SPECIFICHE TECNICHE WEB SERVICES	04	28/05/13	APPROVATO	C .COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO	L. DELLANGELO
2.1	SPECIFICHE TECNICHE WEB SERVICES	05	16/09/13	APPROVATO	C. COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO	L. DELLANGELO
2.1	SPECIFICHE TECNICHE WEB SERVICES	06	16/12/13	APPROVATO	C. COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO	L. DELLANGELO
IGFS 5.2 (E SUCCESSIVE) BUY NOW 2.2	SPECIFICHE TECNICHE WEB SERVICES	07	07/10/14	APPROVATO	C. COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO	L. DELLANGELO
IGFS 5.2 (E SUCCESSIVE) BUY NOW 2.2	SPECIFICHE TECNICHE WEB SERVICES	08	03/11/14	APPROVATO	C. COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO	L. DELLANGELO
IGFS 5.2 (E SUCCESSIVE) BUY NOW 2.2	SPECIFICHE TECNICHE WEB SERVICES	09	24/11/14	APPROVATO	C. COLOMBO - L. MARINO M. SERRA - S. DI LUZIO - M. PEZZO - M. PEDRINAZZI	L. DELLANGELO
NOTE	Modifiche Ed 09: inserito Par. 3.6 INTEGRAZIONE API CON CMS e par. 3.6.1 MAGENTO. Ns. Rif.: igfs_buynow_webserv-api_spt09.pdf					
 Networks & Transactional Systems SpA Strada 4 Palazzo A5 MilanoFiori - 20090 Assago MI - Tel. +39 02822765.1 - Fax +39.0257514170						
TUTTE LE INFORMAZIONI RIPORTATE NEL PRESENTE DOCUMENTO SONO CONFIDENZIALI E NON POSSONO ESSERE UTILIZZATE INTERAMENTE O PARZIALMENTE SENZA L'AUTORIZZAZIONE SCRITTA DA PARTE DI N&TS SPA						

SOMMARIO

PREFAZIONE	2
INTRODUZIONE	5
1 DESCRIZIONE DEL PROCESSO.....	6
1.1 Registrazione utente sul sito dell' esercente.....	6
1.2 Inizializzazione richiesta di pagamento	6
1.3 Pagina di pagamento.....	9
1.4 Notifica all' esercente.....	10
1.5 Funzionalità batch	12
1.5.1 Invio file.....	12
1.5.2 Acquisizione file esiti.....	13
2 WEB SERVICES	14
2.1 PaymentInitGateway.wsdl (Pagamenti online)	14
2.1.1 Metodo init().....	14
2.1.2 Metodo verify().....	17
2.1.3 Metodo selector().....	19
2.2 PaymentTranGateway.wsdl (Pagamenti diretti per carte di credito).....	21
2.2.1 Metodo auth()	21
2.2.2 Metodo confirm().....	24
2.2.3 Metodo voidAuth()	26
2.2.4 Metodo credit().....	28
2.3 BatchGateway.wsdl (Funzionalità batch).....	30
2.3.1 Metodo submit().....	30
2.3.2 Metodo fetch().....	32
2.4 MPIGateway.wsdl (Funzionalità MPI)	34
2.4.1 Metodo enroll().....	34
2.4.2 Metodo auth()	37
2.5 Esempio di implementazione Pagamento online	39
2.5.1 WS Init	39
2.5.2 WS Verify.....	41
3 API.....	43
3.1 Pagamenti online.....	43
3.1.1 Classe IgfsCgInit.....	43
3.1.2 Classe IgfsCgVerify	45
3.1.3 Classe IgfsCgSelector	47
3.2 Pagamenti diretti per carte di credito.....	49

3.2.1	Classe IgfsCgAuth	49
3.2.2	Classe IgfsCgConfirm	52
3.2.3	Classe IgfsCgVoidAuth	54
3.2.4	Classe IgfsCgCredit	56
3.3	Funzionalità batch	58
3.3.1	Classe IgfsBatchSubmit	58
3.3.2	Classe IgfsBatchFetch	59
3.4	Funzionalità MPI	61
3.4.1	Classe IgfsCgMpiEnroll	61
3.4.2	Classe IgfsCgMpiAuth	63
3.5	Esempi di implementazione con API	65
3.5.1	Pagamenti online	65
3.5.1.1	Java Init	65
3.5.1.2	Java Verify	67
3.5.1.3	.NET Init	69
3.5.1.4	.NET Verify	71
3.5.1.5	PHP Init	73
3.5.1.6	PHP Verify	74
3.5.2	Pagamenti diretti per carte di credito	75
3.5.2.1	Java Auth	75
3.5.2.2	Java Confirm	77
3.5.2.3	Java Void	78
3.5.2.4	Java Credit	79
3.5.2.5	.NET Auth	80
3.5.2.6	.NET Confirm	82
3.5.2.7	.NET Void	83
3.5.2.8	.NET Credit	84
3.5.3	Funzionalità batch	85
3.5.3.1	Java Submit	85
3.5.3.2	Java Fetch	86
3.5.3.3	.NET Submit	87
3.5.3.4	.NET Fetch	88
3.5.4	Funzionalità MPI	89
3.5.4.1	Java MPI Enroll	89
3.5.4.2	Java MPI Auth	91
3.5.4.3	.NET MPI Enroll	92
3.5.4.4	.NET MPI Auth	94
3.6	Integrazione API con CMS	95
3.6.1	Magento	95
APPENDICE		98
APPENDICE A: CALCOLO SIGNATURE		98
APPENDICE B: FILE BATCH		100
APPENDICE C: CODICI RITORNO		103

INTRODUZIONE

Nel presente documento vengono illustrate le specifiche tecniche dei Web Services, e delle API, necessari per l'integrazione nel proprio portale web di e-commerce, con il sistema di pagamento elettronico **IGFS®** opzione **BUY NOW** di N&TS.

Tale opzione è integrata al prodotto base IGFS® opzione PA-DSS.

In merito agli standard PA-DSS (PCI) è importante premettere che:

- N&TS non si avvale in alcun caso di distributori o terzi integratori, per cui nessun riferimento a queste figure sarà fatto nel resto del documento e quindi nessuna indicazione al loro operato sarà fatta nella guida.
- Gli unici dati sensibili memorizzati dai programmi N&TS sono il PAN carta e la Data scadenza della carta. Non sono invece rilevanti per i programmi in oggetto di certificazione i Nomi dei titolari della carta, il Codice di servizio, né sono memorizzati i dati completi della striscia magnetica e i CAV2/CID/CVC2/CVV2, come non lo sono il PIN e il PIN Block.

Per ogni documentazione riguardo gli standard PA-DSS si faccia riferimento al sito:

<https://www.pcisecuritystandards.org/>

Il prodotto base IGFS® con opzione PA-DSS è composto dal **modulo IGFS®** per la gestione dei pagamenti e dall'opzione **IGFS® MPI** (nel resto del documento indicato come MPI), per la gestione del protocollo 3D-Secure; entrambi sviluppati sotto il framework JTMS® di N&TS.

1 DESCRIZIONE DEL PROCESSO

1.1 Registrazione utente sul sito dell'esercente

I portali web, che offrono servizi di e-commerce, prevedono solitamente una fase iniziale di registrazione in cui richiedono al cliente dati di carattere anagrafico, uniti a riferimenti di posta elettronica e recapiti di domicilio e/o telefonici.

La fase di registrazione, permette di censire l'utente associandogli un profilo web corrispondente e, successivamente, di garantire un accesso, previa autenticazione, a tutela dei dati del cliente. Dopo essersi autenticato, il cliente può iniziare la navigazione sul sito di e-commerce selezionando i prodotti di proprio interesse e popolando il carrello elettronico (shopping-cart) con la lista dei prodotti da ordinare.

1.2 Inizializzazione richiesta di pagamento

Ultimata la fase di selezione dei prodotti, il cliente viene veicolato su una pagina di conferma dell'acquisto. In questa pagina, dopo aver selezionato lo strumento da utilizzare per il pagamento, viene invitato a premere il pulsante relativo alla funzione di "acquisto/compra" o di uguale significato.

Ricevuta la conferma di acquisto, l'esercente invia al gateway IGFS la richiesta pagamento, attraverso il metodo `init()` del servizio descritto nel wsdl `PaymentInitGateway.wsdl` o attraverso le API N&TS (classe `IgfsCgInit`).

Il messaggio contiene le seguenti proprietà obbligatorie:

- Identificativo dell'esercente: `tid` (fornito dall'amministratore di IGFS)
- Chiave per la firma digitale della richiesta: `ksig` (fornito dall'amministratore di IGFS)
- Tipo di Transazione: `trType` (Pre Autorizzazione AUTH o Autorizzazione a livello contabile PURCHASE).
- Identificativo dell'ordine: `shopID`.
- Identificativo del cliente: `shopUserRef` (es. email dell'utente)
- Importo della Transazione: `amount`.
- Valuta della Transazione: `currencyCode`.
- Identificativo della lingua con la quale verrà visualizzata la pagina di pagamento: `langID`.

- URL http di errore dell'esercente (pagina di errore definita dall'esercente verso cui IGFS deve eseguire la redirect del browser del cliente, a fronte di anomalie/errori durante il flusso di pagamento): **errorUrl**.
- URL http di notifica dell'esercente (pagina di notifica definita dall'esercente verso cui IGFS deve eseguire una richiesta http/s informando l'esercente che i dati di relativi all'esito della transazione di pagamento sono disponibili): **notifyUrl**

E' possibile inoltre inserire fino a 5 informazioni proprietarie attraverso le seguenti proprietà: **addInfo1**, **addInfo2**, **addInfo3**, **addInfo4**, **addInfo5**. Queste verranno memorizzate sulla base dati e quindi legate all'ordine.

Al completamento della fase di inizializzazione, IGFS restituisce all'esercente le seguenti informazioni che si differenziano in base all'esito applicativo:

esito positivo:

- Identificativo della richiesta di pagamento (proprietà **paymentID**), generato da IGFS, che dovrà essere utilizzato nelle successive operazioni.
- URL per l'instradamento (proprietà **redirectURL**) verso la pagina per l'inserimento dei dati dello strumento di pagamento.

esito negativo:

- descrizione dell'errore rilevato da IGFS.
Ad esempio, in caso di invio di una richiesta con un campo mancante IGFS risponde con l'esito **IGFS_20000** (proprietà **rc**) e la descrizione **"Missing shopUserRef"** (proprietà **errorDesc**).

Per un elenco completo delle codifiche degli esiti previsti si veda *Appendice C*.

Utilizzando le API N&TS, le suddette informazioni sono fruibili utilizzando gli opportuni metodi **"getter"** della stessa.

Ad esempio la descrizione dell'errore, si può ottenere con l'invocazione del metodo **getErrorDesc()**, mentre l'URL per la redirect del browser del cliente sulla pagina "buy now", attraverso il metodo **getRedirectURL()**.

Se l'inizializzazione è stata superata con successo l'esercente effettua una redirect del browser Internet del cliente verso la pagina "buy now" di IGFS il cui URL è fruibile attraverso i Web Services N&TS.

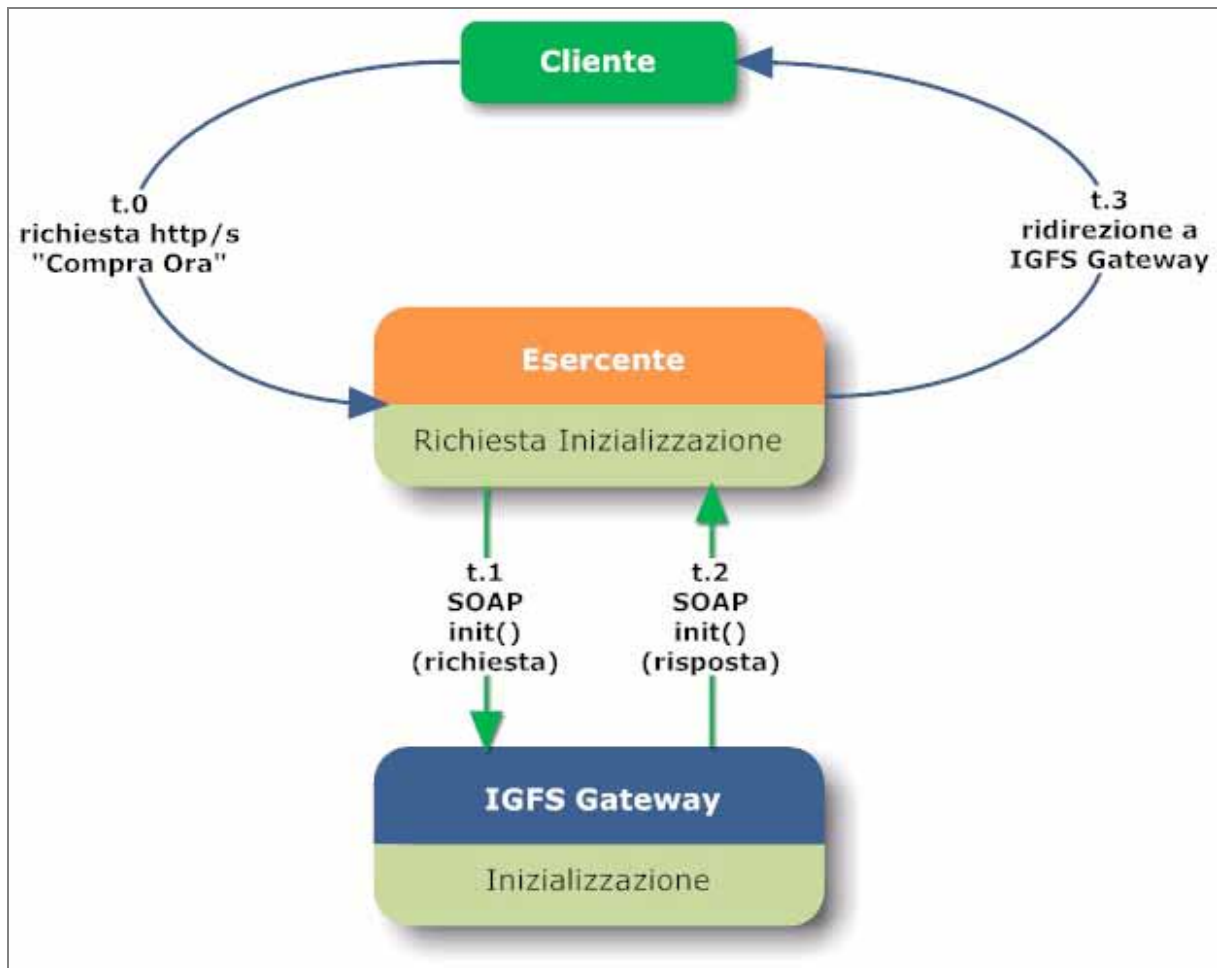


Fig. 1 – Inizializzazione Richiesta di Pagamento

1.3 Pagina di pagamento

Il cliente, per completare l'operazione di pagamento, inserisce i dati richiesti nella pagina web proposta dal modulo buy-now.

LOGO

H E A D E R

PAGAMENTO CON CARTA DI CREDITO

Inserisci dati
In questa pagina devi inserire i dati relativi alla tua carta di credito per poter effettuare il pagamento.

Acquisto presso	Merchant di Test
Riferimento	-1011191468496102996
Carte accettate	
Importo	0,01 EUR
Numero carta*	<input type="text"/>
Scadenza*	04 - 2011
Codice di controllo*	<input type="text"/> Visualizza la posizione del codice CW2/CVC2/4BCD sulla carta
Indirizzo email	cliente@email.it

(*) Campi obbligatori

F O O T E R

Fig. 2 – Pagina di inserimento Dati di pagamento con Carta di Credito (BUY NOW)

NOTA: Tale pagina può essere personalizzata graficamente dal merchant.

1.4 Notifica all'esercente

Al termine del pagamento, IGFS ridirige il cliente verso l'URL di notifica che lo stesso esercente ha comunicato a IGFS in fase di inizializzazione.

A fronte di questa richiesta, per acquisire le informazioni relative all'esito della transazione di pagamento, l'esercente deve invocare il metodo `verify()` del servizio descritto nel wsdl `PaymentInitGateway.wsdl`.

Nel caso in cui vengano utilizzate le API, questa operazione avviene attraverso la classe `IgfsCgVerify`

Ricevuti i dati relativi all'esito della transazione, l'esercente dovrà redirigere il cliente verso la pagina di riepilogo.

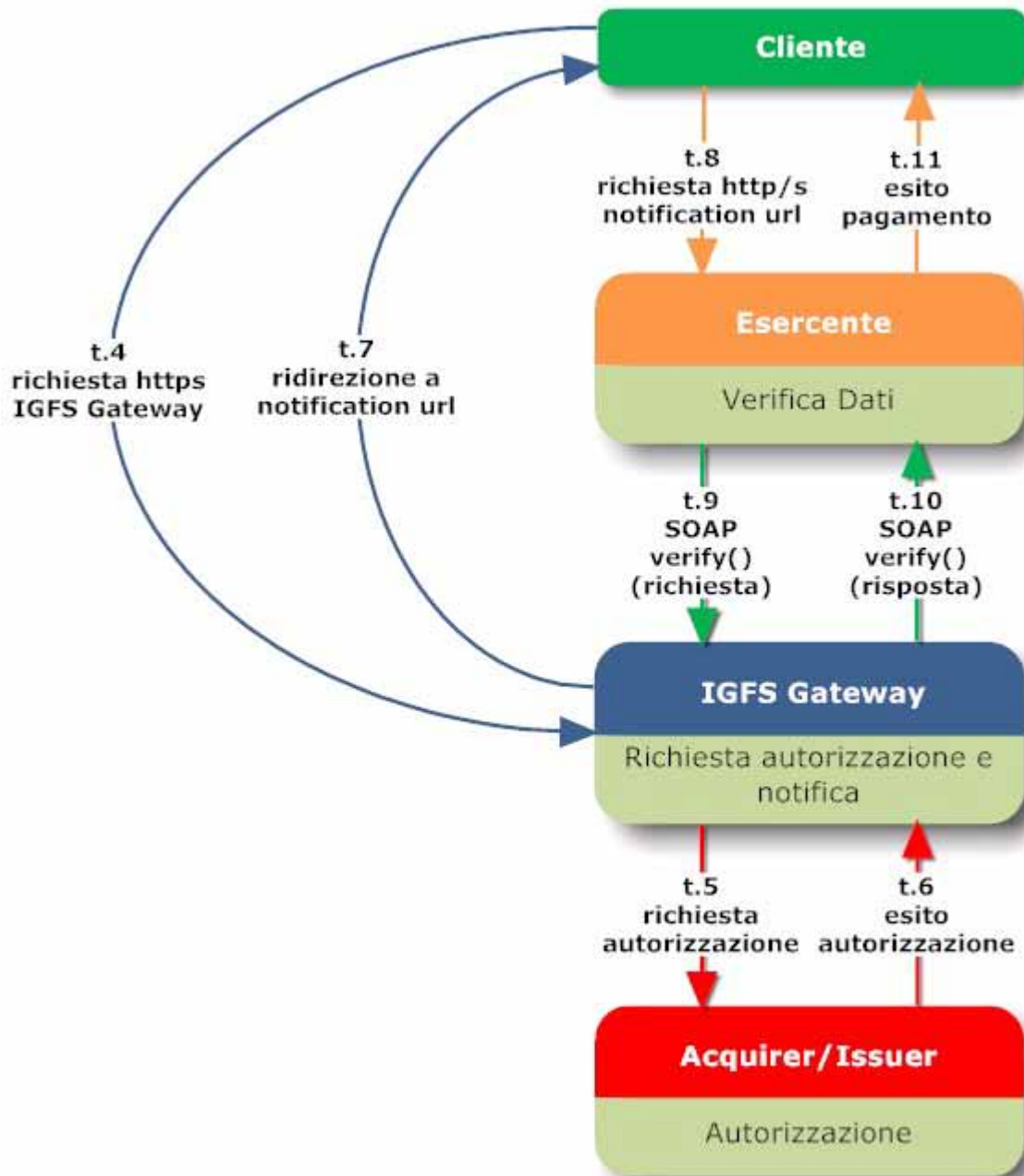


Fig. 3 – Autorizzazione e Verifica Richiesta di Pagamento

Il flusso delle operazioni per la transazione di pagamento è terminato e il cliente può proseguire con la navigazione sul portale dell'esercente.

1.5 Funzionalità batch

Il perfezionamento dell'acquisto (addebito al titolare), il suo annullamento (ripristino del plafond di spesa) o la restituzione degli importi addebitati per acquisti già perfezionati (accredito al titolare) può essere effettuato anche in modalità batch.

Tale modalità, consigliata per richieste massive, prevede:

- la sottomissione a IGFS di un file contenente le informazioni necessarie per l'individuazione delle transazioni e delle operazioni da effettuare su di esse;
- l'attesa del completamento della elaborazione del file sottomesso;
- il recupero del file contenente gli esiti delle operazioni richieste.

Nel caso in cui le operazioni online siano state effettuate specificando l'addebito contestuale all'autorizzazione (PURCHASE) l'unica operazione permessa è il rimborso (accredito al titolare).

Il formato dei file da inviare e da ricevere vengono descritti nell'*Appendice B*.

1.5.1 Invio file

La richiesta di invio del file viene effettuata al gateway IGFS attraverso il metodo `submit()` del servizio descritto nel wsdl `BatchGateway.wsdl` o attraverso le API N&TS (classe `IgfsCgBatchSubmit`).

Il messaggio contiene le seguenti proprietà obbligatorie:

- Identificativo dell'esercente: `tid`. (fornito dall'amministratore di IGFS)
- Chiave per la firma digitale della richiesta: `ksig` (fornito dall'amministratore di IGFS)
- Identificativo batch per l'esercente : `BatchShopID`
- Contenuto File da inviare: `BatchData`

A completamento della fase di acquisizione file, IGFS risponde all'esercente con informazioni che si differenziano in base all'esito applicativo:

esito positivo:

- Identificativo della richiesta di elaborazione batch (proprietà `batchID`), generato da IGFS.

esito negativo:

- Descrizione dell'errore rilevato da IGFS.
Per un elenco completo delle codifiche degli esiti previsti si veda *Appendice C*.

1.5.2 Acquisizione file esiti

La richiesta di acquisizione del file con gli esiti viene effettuata al gateway IGFS attraverso il metodo `fetch()` del servizio descritto nel wsdl `BatchGateway.wsdl` o attraverso le API N&TS (classe `IgfsBatchFetch`).

Il messaggio contiene le seguenti proprietà obbligatorie:

- Identificativo dell'esercente: `tid`.
- Chiave per la firma digitale della richiesta : `ksig`
- Identificativo batch per il quale si richiede l'esito: `BatchShopID`

A completamento della fase di acquisizione file, IGFS risponde all'esercente con informazioni che si differenziano in base all'esito applicativo:

esito positivo:

- Stato della richiesta di elaborazione batch (proprietà `status`);
Può assumere uno dei seguenti valori:
 - `NOT_PROCESSED`
 - `PROCESSING`
 - `PROCESSED`
 - `ERROR`

esito negativo:

- descrizione dell'errore rilevato da IGFS.
Per un elenco completo delle codifiche degli esiti previsti si veda *Allegato A*.

Nello stato `NOT_PROCESSED` o `PROCESSING` l'elaborazione non risulta ancora completata ed è quindi necessario attendere e procedere a una successive interrogazione sino al completamento con successo (`PROCESSED`) o con errori (`ERROR`).

In caso di completamento positivo le proprietà `size` e `batchData` conterranno, rispettivamente, le dimensioni e il contenuto del file esiti.

2 WEB SERVICES

2.1 PaymentInitGateway.wsdl (Pagamenti online)

2.1.1 Metodo init()

Il metodo:

```
PaymentInitResponse init(PaymentInitRequest request);
```

del servizio `PaymentInitGateway` viene utilizzato per eseguire una inizializzazione della richiesta di pagamento.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentInitRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: <ul style="list-style-type: none">- Tid- ShopID- ShopUserRef- ShopUserName- ShopUserAccount- TrType- Amount- CurrencyCode- LangID- NotifyURL- ErrorURL- AddInfo1- AddInfo2- AddInfo3- AddInfo4- AddInfo5 Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento

String[256]	ShopUserRef Identificativo cliente (es:email)
String[256]	ShopUserName Cognome e Nome del cliente (separati dal carattere ,) (es. rossi,mario)
String[64]	ShopUserAccount Account cliente del portale merchant
PURCHASE, AUTH, VERIFY	TrType Tipologia della richiesta
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta
IT, EN	LangID Codice iso 639-2 relativo alla pagina di inserimento dei dati di pagamento
URL[512]	NotifyURL URL relativo alla pagina di notifica esito
URL[512]	ErrorURL URL relativo alla pagina di errore
String[256]	AddInfo1 Campo a disposizione dell'esercente
String[256]	AddInfo2 Campo a disposizione dell'esercente
String[256]	AddInfo3 Campo a disposizione dell'esercente
String[256]	AddInfo4 Campo a disposizione dell'esercente
String[256]	AddInfo5 Campo a disposizione dell'esercente
String[100]	Description Causale di pagamento
Boolean	Recurrent Pagamento ricorrente
String[268]	FreeText Testo libero

PaymentInitResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
String[32]	PaymentID Codice paymentID associato alla richiesta
URL[512]	RedirectURL Url associato alla pagina di "buynow"

2.1.2 Metodo verify()

Il metodo:

```
PaymentVerifyResponse verify(PaymentVerifyRequest request);
```

del servizio `PaymentInitGateway` viene utilizzato per eseguire una operazione di verifica dati della richiesta di pagamento.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentVerifyRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - ShopID - PaymentID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[32]	PaymentID Codice paymentID associato alla richiesta

PaymentVerifyResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
Long[16]	TranID Codice Ordine processato
String[32]	AuthCode Codice di autorizzazione restituito dall' issuer
String[1]	EnrStatus Stato di iscrizione carta al servizio 3D Secure
String[1]	AuthStatus Esito autenticazione carta al servizio 3D Secure
String[32]	Brand Brand carta di credito es. (VISA, MASTERCARD,...)
String[19]	MaskedPan Numero di carta mascherato (se attivo nel profilo terminale)
Integer[2]	ExpireMonth Mese di scadenza (se attivo nel profilo terminale)
Integer[4]	ExpireYear Anno di scadenza (se attivo nel profilo terminale)

2.1.3 Metodo selector()

Il metodo:

`PaymentSelectorResponse selector(PaymentSelectorRequest request);`

del servizio `PaymentInitGateway` viene utilizzato per ottenere la lista dei possibili strumenti di pagamento legati al selettore richiesto.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentSelectorRequest	
Tipo[Dimensione]	Property
String	<p>Signature Firma del messaggio composta dalla concatenazione dei campi:</p> <ul style="list-style-type: none"> - Tid - ShopID - ShopUserRef - TrType - Amount - CurrencyCode - LangID - AddInfo1 - AddInfo2 - AddInfo3 - AddInfo4 - AddInfo5 <p>Per il calcolo della firma si veda l'APPENDICE A.</p>
String[16]	<p>Tid Codice terminale dell' esercente</p>
String[256]	<p>ShopID Chiave esterna identificante il pagamento</p>
String[256]	<p>ShopUserRef Identificativo cliente (es:email)</p>
PURCHASE, AUTH, VERIFY	<p>TrType Tipologia della richiesta</p>
Long[12]	<p>Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)</p>
EUR, USD	<p>CurrencyCode Valuta</p>
IT, EN	<p>LangID Codice iso 639-2 relativo alla pagina di inserimento dei dati di pagamento</p>

String[256]	AddInfo1 Campo a disposizione dell' esercente
String[256]	AddInfo2 Campo a disposizione dell' esercente
String[256]	AddInfo3 Campo a disposizione dell' esercente
String[256]	AddInfo4 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente

PaymentSelectorResponse

Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell' errore/anomalia
TerminalInfo[]	Terminal Terminali abilitati al pagamento (0..*)

TerminalInfo

Tipo[Dimensione]	Property
String[16]	Tid Codice terminale dell' esercente
String[256]	Description Descrizione codice terminale dell' esercente
String[2]	PayInstr Codice della modalità di pagamento
String[64]	PayInstrDescription Descrizione della modalità di pagamento (Label del bottone)
URL[512][[]]	ImgUrl Url associato all' immagine (0..*)

2.2 PaymentTranGateway.wsdl (Pagamenti diretti per carte di credito)

2.2.1 Metodo auth()

Il metodo:

```
PaymentAuthResponse auth(PaymentAuthRequest request);
```

del servizio `PaymentTranGateway` viene utilizzato per eseguire una richiesta di autorizzazione diretta su carta di credito.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentAuthRequest	
Tipo[Dimensione]	Property
String	<p>Signature Firma del messaggio composta dalla concatenazione dei campi:</p> <ul style="list-style-type: none"> - Tid - ShopID - ShopUserRef - ShopUserName - ShopUserAccount - ShopUserIP - TrType - Amount - CurrencyCode - Pan - Cvv2 - ExpireMonth - ExpireYear - AddInfo1 - AddInfo2 - AddInfo3 - AddInfo4 - AddInfo5 <p>Per il calcolo della firma si veda l'<i>APPENDICE A</i>.</p>
String[16]	<p>Tid Codice terminale dell'esercente</p>
String[256]	<p>ShopID Chiave esterna identificante il pagamento</p>

String[256]	ShopUserRef Identificativo cliente (es:email)
String[256]	ShopUserName Cognome e Nome del cliente (separati dal carattere ,) (es. rossi,mario)
String[64]	ShopUserAccount Account cliente del portale merchant
String[256]	ShopUserIP Indirizzo IP cliente
PURCHASE, AUTH, VERIFY	TrType Tipologia della richiesta
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta
String[19]	Pan Numero di carte
String[4]	Cvv2 Numero di sicurezza sul retro della carta
Integer[2]	ExpireMonth Mese di scadenza
Integer[4]	ExpireYear Anno di scadenza
String[256]	AccountName Cognome e Nome del titolare (separati dal carattere ,) (es. bianchi,gino)
String[256]	AddInfo1 Campo a disposizione dell' esercente
String[256]	AddInfo2 Campo a disposizione dell' esercente
String[256]	AddInfo3 Campo a disposizione dell' esercente
String[256]	AddInfo4 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente
String[1]	EnrStatus Dato ricevuto in risposta dall' MPI
String[1]	AuthStatus Dato ricevuto in risposta dall' MPI
String[28]	Cavv Dato ricevuto in risposta dall' MPI

String[28]	Xid Dato ricevuto in risposta dall'MPI
String[100]	Description Causale di pagamento
Boolean	Recurrent Pagamento ricorrente
String[268]	FreeText Testo libero

PaymentAuthResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[32]	AuthCode Codice di autorizzazione restituito dall' issuer
String[32]	Brand Brand carta di credito es. (VISA, MASTERCARD,...)

2.2.2 Metodo confirm()

Il metodo:

```
PaymentConfirmResponse confirm(PaymentConfirmRequest request);
```

del servizio `PaymentTranGateway` viene utilizzato per movimentare una autorizzazione effettuata con carta di credito.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentConfirmRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - ShopID - Amount - RefTranID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da movimentare
Boolean	SplitTran è true se la conferma è parziale

PaymentConfirmResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente
Long[12]	PendingAmount Eventuale importo non confermato

2.2.3 Metodo voidAuth()

Il metodo:

```
PaymentVoidAuthResponse voidAuth(PaymentVoidAuthRequest request);
```

del servizio `PaymentTranGateway` viene utilizzato per stornare una autorizzazione effettuata con carta di credito.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentVoidAuthRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - ShopID - Amount - RefTranID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da annullare

PaymentVoidAuthResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente

2.2.4 Metodo credit()

Il metodo:

```
PaymentCreditResponse credit(PaymentCreditRequest request);
```

del servizio `PaymentTranGateway` viene utilizzato per riaccreditare una autorizzazione effettuata con carta di credito.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

PaymentCreditRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - ShopID - Amount - RefTranID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da riaccreditare
Boolean	SplitTran è true se la conferma è parziale

PaymentCreditResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente

2.3 BatchGateway.wsdl (Funzionalità batch)

Il modulo batch è basato sulle specifiche MTOM (<http://www.w3.org/TR/soap12-mtom/>) pertanto è necessario utilizzare un Client SOAP compatibile con tale specifica (Apache Axis 2, Apache CXF, Microsoft .Net); nel caso in cui ciò non fosse possibile si raccomanda l'utilizzo delle API N&TS.

2.3.1 Metodo submit()

Il metodo:

```
BatchSubmitResponse submit(BatchSubmitRequest request);
```

del servizio **BatchGateway** viene utilizzato per inviare il file di conferma, ovvero le azione da intraprendere sulle transazioni specificate (conferma, storno, credito).

Di seguito l'elenco delle proprietà della richiesta e della risposta.

BatchSubmitRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - BatchShopID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell' esercente
String[256]	BatchShopID Codice Batch associato alla richiesta
Byte[]	BatchData Contenuto File conferme

BatchSubmitResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
String[18]	BatchID Identificatore Batch

2.3.2 Metodo fetch()

Il metodo:

```
BatchFetchResponse fetch(BatchFetchRequest request);
```

del servizio `BatchGateway` viene utilizzato per prelevare il file di risposta al file conferme inviato tramite il metodo `submit()`.

Di seguito l'elenco delle proprietà della richiesta e della risposta.

BatchFetchRequest	
Tipo[Dimensione]	Property
String	Signature Firma del messaggio composta dalla concatenazione dei campi: - Tid - BatchShopID Per il calcolo della firma si veda l' <i>APPENDICE A</i> .
String[16]	Tid Codice terminale dell' esercente
String[256]	BatchShopID Codice Batch associato alla richiesta

BatchFetchResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
String[32]	Status Stato elaborazione Batch Possibili valori: <i>NOT_PROCESSED</i> <i>PROCESSING</i> <i>PROCESSED</i> <i>ERROR</i>
Long	Size Dimensione del file risposta
Byte[]	BatchData Contenuto File esiti

2.4 MPIGateway.wsdl (Funzionalità MPI)

2.4.1 Metodo enroll()

Il metodo:

```
MPIEnrollResponse enroll(MPIEnrollRequest request);
```

del servizio **MPIGateway** serve a verificare se la carta è iscritta al servizio **3D Secure**

Di seguito l'elenco delle proprietà della richiesta e della risposta.

MPIEnrollRequest	
Tipo[Dimensione]	Property
String	<p>Signature Firma del messaggio composta dalla concatenazione dei campi:</p> <ul style="list-style-type: none"> - Tid - ShopID - ShopUserRef - Amount - CurrencyCode - Pan - ExpireMonth - ExpireYear - TermUrl - Description - AddInfo1 - AddInfo2 - AddInfo3 - AddInfo4 - AddInfo5 <p>Per il calcolo della firma si veda l'<i>APPENDICE A</i>.</p>
String[16]	<p>Tid Codice terminale dell' esercente</p>
String[256]	<p>ShopID Chiave esterna identificante il pagamento</p>
String[256]	<p>ShopUserRef Identificativo cliente (es:email)</p>
Long[12]	<p>Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)</p>
EUR, USD	<p>CurrencyCode Valuta</p>

String[19]	Pan Numero di carte
Integer[2]	ExpireMonth Mese di scadenza
Integer[4]	ExpireYear Anno di scadenza
URL[512]	TermURL URL dove ACS ridirige a fine del processo di Auth
String[100]	Description Valore riportato nella form di ACS
String[256]	AddInfo1 Campo a disposizione dell'esercente
String[256]	AddInfo2 Campo a disposizione dell'esercente
String[256]	AddInfo3 Campo a disposizione dell'esercente
String[256]	AddInfo4 Campo a disposizione dell'esercente
String[256]	AddInfo5 Campo a disposizione dell'esercente

MPIEnrollResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
String[28]	Xid Identificativo della transazione
String[1]	EnrStatus Y - Authentication Available Yes No N - Cardholder Not Enrolled No Yes U - Unable to Authenticate No Yes E - any error message here
String	PaReq Process auth response
String	Md Merchant Data
URL[512]	AcsURL URL dall'ACS dove il browser deve essere rediretto E' necessario aggiungere i seguenti parametri all URL ricevuta - PaReq - MD - TermURL
String	AcsPage Codice HTML da inviare al Browser per arrivare alla pagine ACS

2.4.2 Metodo auth()

Il metodo:

```
MPIAuthResponse auth(MPIAuthRequest request);
```

del servizio **MPIGateway** serve a verificare il risultato dell'autenticazione **3D Secure** del titolare

Di seguito l'elenco delle proprietà della richiesta e della risposta.

MPIAuthRequest	
Tipo[Dimensione]	Property
String	<p>Signature Firma del messaggio composta dalla concatenazione dei campi:</p> <ul style="list-style-type: none"> - Tid - ShopID - PaRes - Md <p>Per il calcolo della firma si veda l'<i>APPENDICE A</i>.</p>
String[16]	<p>Tid Codice terminale dell'esercente</p>
String[256]	<p>ShopID Chiave esterna identificante il pagamento</p>
String	<p>PaRes Process auth response (da msg enrollment)</p>
String	<p>Md Merchant Data</p>

MPIAuthResponse	
Tipo[Dimensione]	Property
Boolean	Error Restituisce true in presenza di un errore/anomalia
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
String[28]	Xid Identificativo della transazione
String[1]	AuthStatus Y - Authenticated N - Cardholder Not Authenticated U - Unable to Authenticate E - any error
String[28]	Cavv CardHolder Auth. Verification Value
String[8]	Eci Electronic Commerce Indicator

2.5 Esempio di implementazione Pagamento online

Al fine di facilitare la comprensione dei metodi relativi all'utilizzo dei Web Services sopra descritti, riportiamo di seguito, a titolo di esempio, le operazioni applicative necessarie per la sottomissione di una richiesta di pagamento verso il gateway IGFS.

Tali esempi sono basati sul framework Apache CXF e sulle classi client da esso generate:

2.5.1 WS Init

```
<%
// =====
// =          importazione classi di riferimento          =
// =====
%>
<%@page import="import java.io.File" %>
<%@page import="import java.net.MalformedURLException" %>
<%@page import="import java.net.URL" %>
<%@page import="import javax.xml.namespace.QName" %>
<%

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                       =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
URL wsdlURL = new
URL("https://IPGATEWAY/IGFS_CG_SERVICES/services/PaymentInitGatewayPort?wsdl");
QName SERVICE_NAME = new
QName("http://services.api.web.cg.igfs.apps.netsw.it/",
"PaymentInitGateway");

PaymentInitGateway_Service ss = new PaymentInitGateway_Service(wsdlURL,
SERVICE_NAME);
PaymentInitGateway port = ss.getPaymentInitGatewayPort();

String tid          = "123456";
String kSig         = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID      = "5687010820272485455";
String email       = "user@customer.it";
String trType      = "AUTH";
long amount        = 100;
String curCode     = "EUR";
String langID      = "IT";
String errorURL    = "https://merchant/error.jsp";
String notifyURL   = "https://merchant/notify.jsp";

String signature = getSignature(kSig, // KSIGN
```

```

tid, // TID
shopID, // SHOPID
shopUserRef, // SHOPUSERREF
trType, // TRTYPE
amount, // AMOUNT
currencyCode, // CURRENCYCODE
langID, // LANGID
notifyURL, // NOTIFYURL
errorURL); // ERRORURL

Init _init_parameters = new Init();
PaymentInitRequest _init_parametersRequest = new PaymentInitRequest();
_init_parametersRequest.setTid(tid);
_init_parametersRequest.setSignature(signature);
_init_parametersRequest.setShopID(shopID);
_init_parametersRequest.setShopUserRef(shopUserRef);
_init_parametersRequest.setTrType(trType);
_init_parametersRequest.setAmount(amount);
_init_parametersRequest.setCurrencyCode(currencyCode);
_init_parametersRequest.setLangID(langID);
_init_parametersRequest.setNotifyURL(notifyURL);
_init_parametersRequest.setErrorURL(errorURL);
_init_parameters.setRequest(_init_parametersRequest);

// =====
// =          esecuzione richiesta di inizializzazione          =
// =====
InitResponse _init_return = port.init(_init_parameters);

if (_init_return.getResponse().isError()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" +
_init_return.getResponse().getRc() + "&errorDesc=" +
_init_return.getResponse().getErrorDesc());
    return;
}

String paymentID = _init_return.getResponse().getPaymentID();
// NOTA: Salvo il paymentID relativo alla richiesta (es. sul DB)...

// =====
// =          redirect del client verso URL IGFS BuyNow          =
// =====
String redirectURL = _init_return.getResponse().getRedirectURL();
response.sendRedirect(redirectURL.toString());
%>

```


2.5.2 WS Verify

```

<%
// =====
// =          importazione classi di riferimento          =
// =====
%>
<%@page import="import java.io.File" %>
<%@page import="import java.net.MalformedURLException" %>
<%@page import="import java.net.URL" %>
<%@page import="import javax.xml.namespace.QName" %>
<%

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                       =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
URL wsdlURL = new URL("https://IPGATEWAY/IGFS_CG_SERVICES
/services/PaymentInitGatewayPort?wsdl");
QName SERVICE_NAME = new
QName("http://services.api.web.cg.igfs.apps.netsw.it/",
"PaymentInitGateway");

PaymentInitGateway_Service ss = new PaymentInitGateway_Service(wsdlURL,
SERVICE_NAME);
PaymentInitGateway port = ss.getPaymentInitGatewayPort();

String tid          = "123456";
String kSig         = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID       = "5687010820272485455";
String paymentID    = // NOTA: Leggo il paymentID rilasciato in fase di
init (es. dal DB)...
String errorURL     = "https://merchant/error.jsp";
String esitoURL     = "https://merchant/esito.jsp";

String signature = getSignature(kSig, // KSIGN
tid, // TID
shopID, // SHOPID
paymentID); // PAYMENTID

Verify _verify_parameters = new Verify();
PaymentVerifyRequest _verify_parametersRequest = new
PaymentVerifyRequest();
_verify_parametersRequest.setTid(tid);
_verify_parametersRequest.setSignature(signature);
_verify_parametersRequest.setShopID(shopID);
_verify_parametersRequest.setPaymentID(paymentID);
_verify_parameters.setRequest(_verify_parametersRequest);

// =====
// =          esecuzione richiesta di verifica          =
// =====

```

```
VerifyResponse _verify_return = port.verify(_verify_parameters);

if (_verify_return.getResponse().isError()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" +
        _verify_return.getResponse().getRc() + "&errorDesc=" +
        _verify_return.getResponse().getErrorDesc());
    return;
}

// =====
// =          redirect del client verso URL Esito Pagamento Merchant =
// =====
StringBuffer resultUrl = new StringBuffer();
resultUrl.append(esitoURL);
resultUrl.append("?rc=" + verify.getRc());
resultUrl.append("&tranID=" + verify.getTranID());
resultUrl.append("&enrStatus=" + verify.getEnrStatus());
resultUrl.append("&authStatus=" + verify.getAuthStatus());
response.sendRedirect(resultUrl.toString());
%>
```

3 API

3.1 Pagamenti online

3.1.1 Classe IgfsCgInit

La classe `IgfsCgInit` viene utilizzata per eseguire una inizializzazione della richiesta di pagamento.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[256]	ShopUserRef Identificativo cliente
String[256]	ShopUserName Cognome e Nome del cliente (separati dal carattere ,) (es. rossi,mario)
String[64]	ShopUserAccount Account cliente del portale merchant
PURCHASE, AUTH, VERIFY	TrType Tipologia di una richiesta
Long[12]	Amount Importo associato ad una richiesta in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta associata ad una richiesta

IT, EN	LangID Lingua relativa alla pagina di inserimento dei dati sensibili associata ad una richiesta
URL[512]	NotifyURL URL relativo alla pagina di notifica esito di una richiesta
URL[512]	ErrorURL URL relativo alla pagina di errore associata ad una richiesta
String[256]	AddInfo1 Campo a disposizione dell' esercente
String[256]	AddInfo2 Campo a disposizione dell' esercente
String[256]	AddInfo3 Campo a disposizione dell' esercente
String[256]	AddInfo4 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente
String[100]	Description Causale di pagamento
Boolean	Recurrent Pagamento ricorrente
String[268]	FreeText Testo libero

Metodi

Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output

Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia

String[32]	PaymentID Codice paymentID associato ad una richiesta
URL[512]	RedirectURL Url associato alla pagina di "buynow"

3.1.2 Classe IgfsCgVerify

La classe `IgfsCgVerify` viene utilizzata per eseguire una operazione di verifica dati della richiesta di pagamento.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[256]	PaymentID Codice paymentID associato ad una richiesta

Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output

Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
Long[16]	TranID Codice Ordine processato
String[32]	AuthCode Codice di autorizzazione restituito dall' issuer
String[1]	EnrStatus Stato di iscrizione carta al servizio 3D Secure
String[1]	AuthStatus Esito autenticazione carta al servizio 3D Secure
String[8]	Brand Brand carta di credito es. (VISA, MASTERCARD,...)
String[19]	MaskedPan Numero di carta mascherato (se attivo nel profilo terminale)
Integer[2]	ExpireMonth Mese di scadenza (se attivo nel profilo terminale)
Integer[4]	ExpireYear Anno di scadenza (se attivo nel profilo terminale)

3.1.3 Classe IgfsCgSelector

La classe `IgfsCgSelector` viene utilizzata per ottenere la lista dei possibili strumenti di pagamento legati al selettore richiesto.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[256]	ShopUserRef Identificativo cliente
PURCHASE, AUTH, VERIFY	TrType Tipologia di una richiesta
Long[12]	Amount Importo associato ad una richiesta in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta associata ad una richiesta
IT, EN	LangID Lingua relativa alla pagina di inserimento dei dati sensibili associata ad una richiesta
String[256]	AddInfo1 Campo a disposizione dell'esercente
String[256]	AddInfo2 Campo a disposizione dell'esercente
String[256]	AddInfo3 Campo a disposizione dell'esercente
String[256]	AddInfo4 Campo a disposizione dell'esercente
String[256]	AddInfo5 Campo a disposizione dell'esercente

Metodi

Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output

Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
TerminalInfo[]	Terminal Terminali abilitati al pagamento (0..*)

TerminalInfo

Tipo[Dimensione]	Property
String[16]	Tid Codice terminale dell' esercente
String[256]	Description Descrizione codice terminale dell' esercente
String[2]	PayInstr Codice della modalità di pagamento
String[64]	PayInstrDescription Descrizione della modalità di pagamento (Label del bottone)
URL[512][[]]	ImgUrl Url associato all' immagine (0..*)

3.2 Pagamenti diretti per carte di credito

3.2.1 Classe IgfsCgAuth

La classe `IgfsCgAuth` viene utilizzata per eseguire una richiesta di autorizzazione diretta su carta di credito.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[256]	ShopUserRef Identificativo cliente (es:email)
String[256]	ShopUserName Cognome e Nome del cliente (separati dal carattere ,) (es. rossi,mario)
String[64]	ShopUserAccount Account cliente del portale merchant
String[256]	ShopUserIP Indirizzo IP cliente
PURCHASE, AUTH, VERIFY	TrType Tipologia della richiesta
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta
String[19]	Pan Numero di carte
String[4]	Cvv2 Numero di sicurezza sul retro della carta

Integer[2]	ExpireMonth Mese di scadenza
Integer[4]	ExpireYear Anno di scadenza
String[256]	AccountName Cognome e Nome del titolare (separati dal carattere ,) (es. bianchi,gino)
String[256]	AddInfo1 Campo a disposizione dell' esercente
String[256]	AddInfo2 Campo a disposizione dell' esercente
String[256]	AddInfo3 Campo a disposizione dell' esercente
String[256]	AddInfo4 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente
String[1]	EnrStatus Dato ricevuto in risposta dall' MPI
String[1]	AuthStatus Dato ricevuto in risposta dall' MPI
String[28]	Cavv Dato ricevuto in risposta dall' MPI
String[28]	Xid Dato ricevuto in risposta dall' MPI
String[100]	Description Causale di pagamento
Boolean	Recurrent Pagamento ricorrente
String[268]	FreeText Testo libero

Metodi

Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommario Properties Output

Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[32]	AuthCode Codice di autorizzazione restituito dall' issuer
String[8]	Brand Brand carta di credito es. (VISA, MASTERCARD,...)

3.2.2 Classe IgfsCgConfirm

La classe `IgfsCgConfirm` viene utilizzata per movimentare una autorizzazione effettuata con carta di credito.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da movimentare
Boolean	SplitTran è true se la conferma è parziale

Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente
Long[16]	PendingAmount Eventuale importo non confermato

3.2.3 Classe IgfsCgVoidAuth

La classe `IgfsCgVoidAuth` viene utilizzata per stornare una autorizzazione effettuata con carta di credito.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da annullare

Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente

3.2.4 Classe IgfsCgCredit

La classe `IgfsCgCredit` viene utilizzata per riaccreditare una autorizzazione effettuata con carta di credito.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
Long[16]	RefTranID Codice Ordine relativo alla transazione da riaccreditare
Boolean	SplitTran è true se la conferma è parziale

Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
Long[16]	TranID Codice Ordine processato
String[256]	AddInfo1 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo2 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo3 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo4 Dati inviati in fase di autorizzazione dall' esercente
String[256]	AddInfo5 Dati inviati in fase di autorizzazione dall' esercente

3.3 Funzionalità batch

3.3.1 Classe IgfsBatchSubmit

La classe `IgfsBatchSubmit` viene utilizzata per inviare il file di conferma, ovvero le azione da intraprendere sulle transazioni specificate (conferma, storno, credito).

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	BatchShopID Codice Batch associato alla richiesta
File	BatchDataFile Path File conferme

Sommaro Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia
String[18]	BatchID Identificatore Batch

3.3.2 Classe IgfsBatchFetch

La classe `IgfsBatchFetch` viene utilizzata per prelevare il file di risposta al file conferme inviato tramite `IgfsBatchSubmit`.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	BatchShopID Codice Batch associato alla richiesta dall' esercente
File	BatchDataFile Path File esiti

Sommaro Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione di un errore/anomalia

String[32]	Status Stato elaborazione Batch Possibili valori: <i>NOT_PROCESSED</i> <i>PROCESSING</i> <i>PROCESSED</i> <i>ERROR</i>
Long	Size Dimensione del file risposta

3.4 Funzionalità MPI

3.4.1 Classe IgfsCgMpiEnroll

La classe `IgfsCgMpiEnroll` viene utilizzata per verificare se la carta è iscritta al servizio 3D Secure.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell' esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String[256]	ShopUserRef Identificativo cliente (es:email)
Long[12]	Amount Importo in virgola virtuale (es. 100 = 1,00 EUR)
EUR, USD	CurrencyCode Valuta
String[19]	Pan Numero di carte
Integer[2]	ExpireMonth Mese di scadenza
Integer[4]	ExpireYear Anno di scadenza
URL[512]	TermURL URL dove ACS ridirige a fine del processo di Auth
String[100]	Description Valore riportato nella form di ACS
String[256]	AddInfo1 Campo a disposizione dell' esercente
String[256]	AddInfo2 Campo a disposizione dell' esercente

String[256]	AddInfo3 Campo a disposizione dell' esercente
String[256]	AddInfo4 Campo a disposizione dell' esercente
String[256]	AddInfo5 Campo a disposizione dell' esercente

Sommario Metodi

Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommario Properties Output

Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
String[28]	Xid Identificativo della transazione
String[1]	EnrStatus Y - Authentication Available Yes No N - Cardholder Not Enrolled No Yes U - Unable to Authenticate No Yes E - any error message here
String	PaReq Process auth response
String	Md Merchant Data
URL[512]	AcsURL URL dall'ACS dove il browser deve essere rediretto. E' necessario postare i seguenti parametri all URL ricevuta - PaReq - MD - TermURL
String	AcsPage Codice HTML da inviare al Browser per arrivare alla pagine ACS

3.4.2 Classe IgfsCgMpiAuth

La classe `IgfsCgMpiAuth` viene utilizzata per verificare il risultato dell'autenticazione **3D Secure** del titolare.

Sommaro Properties Input	
Tipo[Dimensione]	Property
URL	ServerURL Indirizzo del server di destinazione della richiesta
Integer	Timeout Timeout massimo espresso in millisecondi di completamento di una richiesta
String[64]	KSig Chiave per firmare il messaggio
String[16]	Tid Codice terminale dell'esercente
String[256]	ShopID Chiave esterna identificante il pagamento
String	PaRes Process auth response (da msg enrollment)
String	Md Merchant Data

Sommaro Metodi	
Tipo[Dimensione]	Property
Boolean	execute() Esegue la transazione
	resetFields() Azzeramento parametri di richiesta

Sommaro Properties Output	
Tipo[Dimensione]	Property
String[16]	Rc Esito della richiesta
String[80]	ErrorDesc Descrizione dell'errore/anomalia
String[28]	Xid Identificativo della transazione
String[1]	AuthStatus Y - Authenticated N - Cardholder Not Authenticated U - Unable to Authenticate E - any error
String[28]	Cavv CardHolder Auth. Verification Value
String[8]	Eci Electronic Commerce Indicator

3.5 Esempi di implementazione con API

Al fine di facilitare la comprensione dei metodi relativi all'utilizzo delle API sopra descritte.

3.5.1 Pagamenti online

3.5.1.1 Java Init

```
// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.init.IgfsCgInit" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.init.IgfsCgInit.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.security.SecureRandom" %>
<%@page import="java.util.HashMap" %>
<%@page import="java.util.Map" %>
<%@page import="java.util.Properties" %>
<%@page import="java.io.InputStream" %>

<%
// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                       =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;

String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String email          = "user@customer.it";
TrType trType        = TrType.AUTH;
CurrencyCode curCode = CurrencyCode.EUR;
LangID langID        = LangID.IT;
long amount          = 100;
String errorURL      = "https://merchant/error.jsp";
String notifyURL     = "https://merchant/notify.jsp";

IgfsCgInit init = new IgfsCgInit();
init.setServerURL(new URL(serverURL));
init.setTimeout(timeout);

init.setTid(tid);
init.setKSig(kSig);
init.setShopID(shopID);
init.setShopUserRef(email);
init.setTrType(trType);
init.setCurrencyCode(curCode);
init.setLangID(langID);
init.setAmount(amount);
init.setErrorURL(new URL(errorUrl));
```

```
init.setNotifyURL(new URL(notifyURL));

// =====
// =          esecuzione richiesta di inizializzazione          =
// =====
if (!init.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" + init.getRc() + "&errorDesc=" +
init.getErrorDesc());
    return;
}

String paymentID = init.getPaymentID();
// NOTA: Salvo il paymentID relativo alla richiesta (es. sul DB)...

// =====
// =          redirect del client verso URL IGFS BuyNow          =
// =====
URL redirectURL = init.getRedirectURL();
response.sendRedirect(redirectURL.toString());
%>
```

3.5.1.2 Java Verify

```

<%
// =====
// =          importazione classi di riferimento          =
// =====
%>
<%@page import="it.netsw.apps.igfs.cg.coms.api.init.IgfsCgVerify" %>
<%@page import="java.net.URL" %>
<%@page import="java.security.SecureRandom" %>
<%@page import="java.util.HashMap" %>
<%@page import="java.util.Map" %>
<%@page import="java.util.Properties" %>
<%@page import="java.io.InputStream" %>
<%

// =====
// = impostazione parametri per l'inizializzazione richiesta di          =
// = pagamento.          =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;
String tid           = "123456";
String kSig          = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String paymentID     = // NOTA: Leggo il paymentID rilasciato in fase di
init (es. dal DB)...
String errorURL      = "https://merchant/error.jsp";
String esitoURL      = "https://merchant/esito.jsp";

IgfsCgVerify verify = new IgfsCgVerify();
verify.setServerURL(new URL(serverURL));
verify.setTimeout(timeout);
verify.setTid(tid);
verify.setKSig(kSig);
verify.setShopID(shopID);
verify.setPaymentID(paymentID);

// =====
// =          esecuzione richiesta di verifica          =
// =====
if (!verify.execute()) {
// =====
// = redirect del client su pagina di errore definita dall' esercente =
// =====
response.sendRedirect(errorURL + "?rc=" + verify.getRc() + "&errorDesc="
+ verify.getErrorDesc());
return;
}
// =====

```

```
// =          redirect del client verso URL Esito Pagamento Merchant =  
// =====  
StringBuffer resultUrl = new StringBuffer();  
resultUrl.append(esitoURL);  
resultUrl.append("?rc=" + verify.getRc());  
resultUrl.append("&tranID=" + verify.getTranID());  
resultUrl.append("&enrStatus=" + verify.getEnrStatus());  
resultUrl.append("&authStatus=" + verify.getAuthStatus());  
response.sendRedirect(resultUrl.toString());  
%>
```

3.5.1.3 .NET Init

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.init;

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                       =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String email          = "user@customer.it";
TrType trType        = TrType.AUTH;
CurrencyCode curCode = CurrencyCode.EUR;
LangID langID        = LangID.IT;
long amount          = 100;
String errorURL       = "https://merchant/error.aspx";
String notifyURL      = "https://merchant/notify.ashx";

IgfsCgInit init = new IgfsCgInit();
init.ServerURL = new Uri(serverURL);
init.Timeout = timeout;
init.Tid = tid;
init.KSig = kSig;
init.ShopID = shopID;
init.ShopUserRef = email;
init.TrType = trType;
init.CurrencyCode = curCode;
init.LangID = langID;
init.Amount = amount;
init.ErrorURL = new Uri(errorURL);
init.NotifyURL = new Uri(notifyURL);

// =====
```

```
// =          esecuzione richiesta di inizializzazione          =
// =====
if (!init.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Response.Redirect(errorURL + "?rc=" + init.Rc + "&errorDesc=" +
init.ErrorDesc);
    return;
}

String paymentID = init.PaymentID;
// NOTA: Salvo il paymentID relativo alla richiesta (es. sul DB)...

// =====
// =          redirect del client verso URL IGFS BuyNow          =
// =====
Uri redirectURL = init.RedirectURL;
Response.Redirect(redirectURL.ToString());
```

3.5.1.4 .NET Verify

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.init;

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                       =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;
String tid           = "123456";
String kSig          = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String paymentID     = // NOTA: Leggo il paymentID rilasciato in fase di
init (es. dal DB)...
String errorURL      = "https://merchant/error.aspx";
String esitoURL      = "https://merchant/esito.aspx";

IgfsCgVerify verify = new IgfsCgVerify();
verify.ServerURL = new Uri(serverURL);
verify.Timeout = timeout;
verify.Tid = tid;
verify.KSig = kSig;
verify.ShopID = shopID;
verify.PaymentID = paymentID;

// =====
// =          esecuzione richiesta di verifica          =
// =====
if (!verify.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Response.Redirect(errorURL + "?rc=" + verify.Rc + "&errorDesc=" +
verify.ErrorDesc);
    return;
}
```

```
}  
  
// =====  
// =          redirect del client verso URL Esito Pagamento Merchant =  
// =====  
StringBuilder resultUrl = new StringBuilder();  
resultUrl.Append(esitoURL);  
resultUrl.Append("?rc=" + verify.Rc);  
resultUrl.Append("&tranID=" + verify.TranID);  
resultUrl.Append("&enrStatus=" + verify.EnrStatus);  
resultUrl.Append("&authStatus=" + verify.AuthStatus);  
Response.Redirect(resultUrl.ToString());
```


3.5.1.5 PHP Init

```

<?php
// =====
// =          importazione classi di riferimento          =
// =====
require('IGFS_CG_API/init/IgfsCgInit.php');

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                    =
// = NB: I parametri riportati sono solo a titolo di esempio        =
// =====
$init = new IgfsCgInit();
$init->serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
$init->timeout = 15000;
$init->tid = "123456";
$init->kSig = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
$init->shopID = "5687010820272485455";
$init->shopUserRef = "user@customer.it";
$init->trType = "AUTH";
$init->currencyCode = "EUR";
$init->amount = 100;
$init->landID = "IT";
$init->notifyURL = "https://merchant/notify.php";
$init->errorURL = "https://merchant/error.php";

// =====
// =          esecuzione richiesta di inizializzazione          =
// =====
if (!$init->execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    header("location:          error.php?rc=".urlencode($init->rc)."&errorDesc=".urlencode($init->errorDesc));
    return;
}

// NOTA: Salvo il $init->paymentID relativo alla richiesta (es. sul DB)...

// =====
// =          redirect del client verso URL IGFS BuyNow          =
// =====
header("location: ".$init->redirectURL);
?>

```

3.5.1.6 PHP Verify

```

<?php
// =====
// =          importazione classi di riferimento          =
// =====
require('IGFS_CG_API/init/IgfsCgVerify.php');

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                    =
// = NB: I parametri riportati sono solo a titolo di esempio        =
// =====
$verify = new IgfsCgVerify();
$verify->serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
$verify->timeout = 15000;
$verify->tid = "123456";
$verify->kSig = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
$verify->shopID = "5687010820272485455";
$verify->paymentID = // NOTA: Leggo il paymentID rilasciato in fase di init
(es. dal DB)...
$errorURL = "https://merchant/error.php";
$esitoURL = "https://merchant/esito.php";

// =====
// =          esecuzione richiesta di verifica          =
// =====
if (!$verify->execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    header("location: ".$errorURL + "?rc=" . $verify->rc . "&errorDesc=" .
$verify->errorDesc);
    return;
}

// =====
// =          redirect del client verso URL Esito Pagamento Merchant    =
// =====
header("location: ".$esitoURL + "?esito=OK&rc=" . $verify->rc . "&tranID="
. $verify->tranID . "&enrStatus=" . $verify->enrStatus . "&authStatus=" .
$verify->authStatus);
?>

```

3.5.2 Pagamenti diretti per carte di credito

3.5.2.1 Java Auth

```
// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.init.IgfsCgAuth" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.init.IgfsCgAuth.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.io.InputStream" %>
<%

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                         =
// = NB: I parametri riportati sono solo a titolo di esempio           =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;
String tid           = "123456";
String kSig          = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String email         = "user@customer.it";
TrType trType        = TrType.AUTH;
CurrencyCode currCode = CurrencyCode.EUR;
long amount          = 100;
String errorURL      = "https://merchant/error.jsp";

// =====
// =          dati rilevati alla carta          =
// =====
String pan            = "451234xxxxxx0112";
int expireMonth      = 1;
int expireYear       = 2018;
String cvv2          = "xxx";

// =====
// =          dati rilevati dalle chiamate MPI          =
// =====
String enrStatus     = "Y";
String authStatus    = "Y";
String cavv          = "123nsdfhfiufksdmlskglk";
String xid           = "121345678912";

IgfsCgAuth auth = new IgfsCgAuth();
auth.setServerURL(new URL(serverURL));
auth.setTimeout(timeout);
auth.setTid(tid);
auth.setKSig(kSig);
```

```
auth.setShopID(shopID);
auth.setShopUserRef(email);
auth.setTrType(trType);
auth.setCurrencyCode(currCode);
auth.setAmount(amount);

auth.setPan(pan);
auth.setExpireMonth(expireMonth);
auth.setExpireYear(expireYear);
auth.setCvv2(cvv2);

auth.setEnrStatus(enrStatus);
auth.setAuthStatus(authStatus);
auth.setCavv(cavv);
auth.setXid(xid);

// =====
// =          esecuzione richiesta di inizializzazione          =
// =====
if (!auth.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" + auth.getRc() + "&errorDesc=" +
auth.getErrorDesc());
    return;
}

// =====
// =          ottengo il risultato e continuo con le operazioni          =
// =====
String result = auth.getRc();
String tranID = auth.getTranID();
String brand = auth.getBrand();
%>
```

3.5.2.2 Java Confirm

```

// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.BaseIgfsCgTran" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgConfirm" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgConfirm.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.io.InputStream" %>
<%
// =====
// = impostazione parametri per l'inizializzazione richiesta di          =
// = confirm.                                                              =
// = NB: I parametri riportati sono solo a titolo di esempio              =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String errorURL      = "https://merchant/error.jsp";
long refTranID       = // Identificativo transazione autorizzata
long amount          = 100;
IgfsCgConfirm confirm = new IgfsCgConfirm();
confirm.setServerURL(new URL(serverURL));
confirm.setTimeout(timeout);
confirm.setTid(tid);
confirm.setKSig(kSig);
confirm.setShopID(shopID);
confirm.setRefTranID(refTranID);
confirm.setAmount(amount);
// =====
// =          esecuzione richiesta          =
// =====
if (!confirm.execute()) {
// =====
// = redirect del client su pagina di errore definita dall' esercente =
// =====
    response.sendRedirect(errorURL + "?rc=" + confirm.getRc() +
"&errorDesc=" + confirm.getErrorDesc());
    return;
}
// =====
// = presa in carico esito transazione per poi proseguire          =
// =====
String result = confirm.getRc();
%>

```

3.5.2.3 Java Void

```

// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.BaseIgfsCgTran" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgVoidAuth" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgVoidAuth.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.io.InputStream" %>
<%
// =====
// = impostazione parametri per l'inizializzazione richiesta di          =
// = void.          =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String errorURL      = "https://merchant/error.jsp";
long refTranID       = // Identificativo transazione autorizzata
long amount          = 100;
IgfsCgVoidAuth voidAuth = new IgfsCgVoidAuth();
voidAuth.setServerURL(new URL(serverURL));
voidAuth.setTimeout(timeout);
voidAuth.setTid(tid);
voidAuth.setKSig(kSig);
voidAuth.setShopID(shopID);
voidAuth.setRefTranID(refTranID);
voidAuth.setAmount(amount);
// =====
// =          esecuzione richiesta          =
// =====
if (!voidAuth.execute()) {
// =====
// = redirect del client su pagina di errore definita dall' esercente =
// =====
    response.sendRedirect(errorURL + "?rc=" + voidAuth.getRc() +
"&errorDesc=" + voidAuth.getErrorDesc());
    return;
}
// =====
// = presa in carico esito transazione per poi proseguire          =
// =====
String result = voidAuth.getRc();
%>

```

3.5.2.4 Java Credit

```

// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.BaseIgfsCgTran" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgCredit" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.tran.IgfsCgCredit.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.io.InputStream" %>
<%
// =====
// = impostazione parametri per l'inizializzazione richiesta di          =
// = credito.                                                              =
// = NB: I parametri riportati sono solo a titolo di esempio              =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID         = "5687010820272485455";
String errorURL       = "https://merchant/error.jsp";
long refTranID        = // Identificativo transazione autorizzata
long amount           = 100;

IgfsCgCredit credit = new IgfsCgCredit();
credit.setServerURL(new URL(serverURL));
credit.setTimeout(timeout);
credit.setTid(tid);
credit.setKSig(kSig);
credit.setShopID(shopID);
credit.setRefTranID(refTranID);
credit.setAmount(amount);

// =====
// =          esecuzione richiesta          =
// =====
if (!credit.execute()) {
// =====
// = redirect del client su pagina di errore definita dall' esercente =
// =====
response.sendRedirect(errorURL + "?rc=" + credit.getRc() +
"&errorDesc=" + credit.getErrorDesc());
return;
}
// =====
// = presa in carico esito credito per poi proseguire          =
// =====
String result = credit.getRc();
%>

```

3.5.2.5 .NET Auth

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.init;

// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = pagamento.                                                    =
// = NB: I parametri riportati sono solo a titolo di esempio        =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String email         = "user@customer.it";
TrType trType        = TrType.AUTH;
CurrencyCode currCode = CurrencyCode.EUR;
long amount          = 100;
String errorURL      = "https://merchant/error.jsp";

// =====
// =          dati rilevati alla carta          =
// =====
String pan            = "451234xxxxxx0112";
int expireMonth      = 1;
int expireYear       = 2018;
String cvv2          = "xxx";

// =====
// =          dati rilevati dalle chiamate MPI          =
// =====
String enrStatus     = "Y";
String authStatus    = "Y";
String cavv          = "123nsdfhfiufksdmlskglk";
String xid           = "121345678912";

IgfsCgAuth auth      = new IgfsCgAuth();
auth.ServerURL       = new Uri(serverURL);
auth.Timeout         = timeout;
auth.Tid             = tid;
auth.KSig            = kSig;
auth.ShopID          = shopID;
auth.ShopUserRef     = email;
auth.TrType          = trType;
```



```
auth.CurrencyCode = currCode;
auth.Amount       = amount;

auth.Pan          = pan;
auth.ExpireMonth  = expireMonth;
auth.ExpireYear   = expireYear;
auth.Cvv2         = cvv2;

auth.EnrStatus    = enrStatus;
auth.AuthStatus   = authStatus;
auth.Cavv         = cavv;
auth.Xid          = xid;

// =====
// =           esecuzione richiesta di inizializzazione           =
// =====
if (!auth.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Response.Redirect(errorURL + "?rc=" + auth.Rc + "&errorDesc=" +
auth.ErrorDesc);
    return;
}

// =====
// =           ottengo il risultato e continuo con le operazioni           =
// =====
String result = auth.Rc;
String tranID = auth.TranID;
String brand  = auth.Brand;
%>
```

3.5.2.6 .NET Confirm

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.tran;
// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = confirm.                                                         =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String errorURL      = "https://merchant/error.aspx";
long refTranID       = // Identificativo transazione autorizzata
long amount          = 100;

IgfsCgConfirm confirm = new IgfsCgConfirm();
confirm.ServerURL = new Uri (serverURL);
confirm.Timeout   = timeout;
confirm.Tid       = tid;
confirm.KSig      = kSig;
confirm.ShopID    = shopID;
confirm.RefTranID = refTranID;
confirm.Amount    = amount;

// =====
// =          esecuzione richiesta          =
// =====
if (!confirm.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Server.Transfer(errorURL + "?rc=" + confirm.Rc + "&errorDesc=" +
confirm.ErrorDesc);
    return;
}
// =====
// = presa in carico esito transazione per poi proseguire          =
// =====
String result = confirm.Rc;
```

3.5.2.7 .NET Void

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.tran;
// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = void          .          =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String errorURL       = "https://merchant/error.aspx";
long refTranID       = // Identificativo transazione autorizzata
long amount          = 100;

IgfsCgVoidAuth voidAuth = new IgfsCgVoidAuth();
voidAuth.ServerURL = new Uri (serverURL);
voidAuth.Timeout   = timeout;
voidAuth.Tid       = tid;
voidAuth.KSig      = kSig;
voidAuth.ShopID    = shopID;
voidAuth.RefTranID = refTranID;
voidAuth.Amount    = amount;

// =====
// =          esecuzione richiesta          =
// =====
if (!voidAuth.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Server.Transfer(errorURL + "?rc=" + voidAuth.Rc + "&errorDesc=" +
voidAuth.ErrorDesc);
    return;
}
// =====
// = presa in carico esito void per poi proseguire          =
// =====
String result = voidAuth.Rc;
```

3.5.2.8 .NET Credit

```

// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.tran;
// =====
// = impostazione parametri per l'inizializzazione richiesta di      =
// = credito.                                                         =
// = NB: I parametri riportati sono solo a titolo di esempio         =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID         = "5687010820272485455";
String errorURL       = "https://merchant/error.aspx";
long refTranID        = // Identificativo transazione autorizzata
long amount           = 100;

IgfsCgCredit credit = new IgfsCgCredit();
credit.ServerURL     = new Uri (serverURL);
credit.Timeout       = timeout;
credit.Tid           = tid;
credit.KSig          = kSig;
credit.ShopID        = shopID;
credit.RefTranID     = refTranID;
credit.Amount        = amount;

// =====
// =          esecuzione richiesta          =
// =====
if (!credit.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Server.Transfer(errorURL + "?rc=" + credit.Rc + "&errorDesc=" +
credit.ErrorDesc);
    return;
}
// =====
// = presa in carico esito credit per poi proseguire          =
// =====
String result = credit.Rc;

```

3.5.3 Funzionalità batch

3.5.3.1 Java Submit

```
// =====
// =          importazione classi di riferimento          =
// =====
import java.io.File;
import java.net.URL;
import it.netsw.apps.igfs.cg.coms.api.batch.*;

// =====
// = impostazione parametri per l'sottomissione file      =
// = NB: I parametri riportati sono solo a titolo di esempio =
// =====
String serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout = 15000;
String tid = "123456";
String kSig = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String batchShopID = "BATCH_1";
File batchDataFile = new File("../files/file.mov");
String batchID = null;

IgfsCgBatchSubmit submit = new IgfsCgBatchSubmit();
submit.setServerURL(new URL(serverURL));
submit.setTimeout(timeout);
submit.setTid(tid);
submit.setKSig(kSig);
submit.setBatchShopID(batchShopID);
submit.setBatchDataFile(batchDataFile);

// =====
// =          esecuzione richiesta di sottomissione      =
// =====
if (!submit.execute()) {
    System.out.println("Submit execute error");
    System.out.println("rc=" + submit.getRc() + " errorDesc=" +
submit.getErrorDesc());
} else {
    batchID = submit.getBatchID();
}
}
```

3.5.3.2 Java Fetch

```
// =====
// =          importazione classi di riferimento          =
// =====
import java.net.URL;
import it.netsw.apps.igfs.cg.coms.api.batch.*;
import it.netsw.apps.igfs.cg.coms.api.batch.IgfsCgBatchFetch.Status;

// =====
// = impostazione parametri per l'sottomissione file      =
// = NB: I parametri riportati sono solo a titolo di esempio =
// =====
String serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout = 15000;
String tid = "123456";
String kSig = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String batchShopID = "BATCH_1";
File batchDataFile = new File("../files/file.res");

IgfsCgBatchFetch fetch = new IgfsCgBatchFetch();
fetch.setServerURL(new URL(serverURL));
fetch.setTimeout(timeout);
fetch.setTid(tid);
fetch.setKSig(kSig);
fetch.setBatchShopID(batchShopID);
fetch.setBatchDataFile(batchDataFile);

// =====
// =          esecuzione richiesta di acquisizione file esiti          =
// =====
if (!fetch.execute()) {
    System.out.println("Fetch execute error. rc= "+ fetch.getRc()+ " - "+
fetch.getErrorDesc());
}

if (fetch.getStatus() == Status.PROCESSED) {
    // Read batchDataFile...
}
}
```

3.5.3.3 .NET Submit

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.init;

// =====
// = impostazione parametri per l'sottomissione file      =
// = NB: I parametri riportati sono solo a titolo di esempio =
// =====
String serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
String tid = "123456";
String kSig = "xHosiSb08fs8BQmt9Yhq3Ub99E8=";
String batchShopID = "batchID1";
String batchDataFile = "c:\data.mov";
String batchID = null;

IgfsCgBatchSubmit submit = new IgfsCgBatchSubmit();
submit.ServerURL = new Uri(serverURL);
submit.Tid = tid;
submit.KSig = kSig;
submit.BatchShopID = batchShopID;
submit.BatchDataFile = batchDataFile;

// =====
// =          esecuzione richiesta di sottomissione      =
// =====
if (!submit.execute())
{
    System.console.WriteLine("Error rc=" + submit.Rc + " errorDesc=" +
submit.ErrorDesc);
}
else
{
    batchID = submit.BatchID;
}
}
```

3.5.3.4 .NET Fetch

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.batch;

// =====
// = impostazione parametri per l'sottomissione file      =
// = NB: I parametri riportati sono solo a titolo di esempio =
// =====
String serverURL = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
String tid = "123456";
String kSig = "xHosiSb08fs8BQmt9Yhq3Ub99E8=";
String batchShopID = "batchID1";
String batchDataFile = "c:\data.res";

IgfsCgBatchFetch fetch = new IgfsCgBatchFetch();
fetch.ServerURL = new Uri(serverURL);
fetch.Tid = tid;
fetch.KSig = kSig;
fetch.BatchShopID = batchShopID;
fetch.BatchDataFile = batchDataFile;

// =====
// =          esecuzione richiesta di acquisizione file esiti      =
// =====
if (!fetch.execute())
{
    System.Console.WriteLine("Error rc=" + fetch.Rc + " errorDesc=" +
fetch.ErrorDesc);
}

if (fetch.Status == Status.PROCESSED)
{
    // Read batchDataFile...
}
```


3.5.4 Funzionalità MPI

3.5.4.1 Java MPI Enroll

```
// =====
// =          importazione classi di riferimento          =
// =====
<%@page import="it.netsw.apps.igfs.cg.coms.api.mpi.BaseIgfsCgMpi" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.mpi.IgfsCgMpiEnroll" %>
<%@page import="it.netsw.apps.igfs.cg.coms.api.mpi.IgfsCgMpiEnroll.*" %>
<%@page import="java.net.URL" %>
<%@page import="java.io.InputStream" %>
<%
// =====
// = impostazione parametri per l'inizializzazione richiesta          =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout          = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String email         = "user@customer.it";
String errorURL      = "https://merchant/error.aspx";
long amount          = 100;
CurrencyCode currCode = CurrencyCode.EUR;
String pan            = "451234xxxxxx0112";
int expireMonth      = 1;
int expireYear       = 2018;
String termURL       = "https://merchant/mpi.jsp";
String shopDesc      = "Acquisto presso merchant XXX";

IgfsCgMpiEnroll enroll = new IgfsCgMpiEnroll();

enroll.setServerURL(new URL(serverURL));
enroll.setTimeout(timeout);
enroll.setTid(tid);
enroll.setKSig(kSig);
enroll.setShopID(shopID);
enroll.setShopUserRef(email);
enroll.setAmount(amount);
enroll.setCurrencyCode(currCode);
enroll.setPan(pan);
enroll.setExpireMonth(expireMonth);
enroll.setExpireYear(expireYear);
enroll.setTermURL(new URL(termURL));
enroll.setDescription(shopDesc);

// =====
// =          esecuzione richiesta verifica iscrizione          =
// =====
```

```
// =====
if (!enroll.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" + enroll.getRc() +
"&errorDesc=" + enroll.getErrorDesc());
    return;
}

// =====
// =                redirect del client verso URL ACS                =
// =====
String enrStatus = enroll.getEnrStatus();
if ("Y".equals(enrStatus)) {
    response.getWriter().print(enroll.getAcsPage());
    return;
}
%>
```

3.5.4.2 Java MPI Auth

```

// =====
// =          importazione classi di riferimento          =
// =====
<@page import="it.netsw.apps.igfs.cg.coms.api.mpi.BaseIgfsCgMpi" %>
<@page import="it.netsw.apps.igfs.cg.coms.api.mpi.IgfsCgMpiAuth" %>
<@page import="it.netsw.apps.igfs.cg.coms.api.mpi.IgfsCgMpiAuth.*" %>
<@page import="java.net.URL" %>
<@page import="java.io.InputStream" %>
<%
// =====
// = impostazione parametri per l'inizializzazione richiesta          =
// = NB: I parametri riportati sono solo a titolo di esempio          =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID         = "5687010820272485455";
String errorURL       = "https://merchant/error.jsp";
String PaRes          = (String) request.getParameter("PaRes");
String md             = (String) request.getParameter("MD");

IgfsCgMpiAuth auth = new IgfsCgMpiAuth();

auth.setServerURL(new URL(serverURL));
auth.setTimeout(timeout);
auth.setTid(tid);
auth.setKSig(kSig);
auth.setShopID(shopID);
auth.setPaRes(PaRes);
auth.setMd(md);

if (!auth.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    response.sendRedirect(errorURL + "?rc=" + auth.getRc() + "&errorDesc="
+ auth.getErrorDesc());
    return;
}

// =====
// = Salvo i dati dell'auth per inviarli nella transazione          =
// =====
String authStatus = auth.getAuthStatus();
String xid = auth.getXid();
String cavv = auth.getCavv();
String eci = auth.getEci();
%>

```

3.5.4.3 .NET MPI Enroll

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.mpi;
// =====
// = impostazione parametri per l'inizializzazione richiesta      =
// = NB: I parametri riportati sono solo a titolo di esempio      =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID         = "5687010820272485455";
String email          = "user@customer.it";
String errorURL       = "https://merchant/error.aspx";
long amount           = 100;
CurrencyCode currCode = CurrencyCode.EUR;
String pan             = "451234xxxxxx0112";
int expireMonth       = 1;
int expireYear        = 2018;
String termURL        = "https://merchant/mpi.jsp";
String shopDesc       = "Acquisto presso merchant XXX";

IgfsCgMpiEnroll enroll = new IgfsCgMpiEnroll();

enroll.ServerURL      = new Uri(serverURL);
enroll.Timeout        = timeout;
enroll.Tid            = tid;
enroll.KSig           = kSig;
enroll.ShopID         = shopID;
enroll.ShopUserRef    = email;
enroll.Amount         = amount;
enroll.CurrencyCode   = currCode;
enroll.Pan            = pan;
enroll.ExpireMonth    = expireMonth;
enroll.ExpireYear     = expireYear;
enroll.TermURL        = new Uri(termURL);
enroll.Description    = shopDesc;

// =====
// =          esecuzione richiesta verifica iscrizione          =
// =====
if (!enroll.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
}
```

```
// =====  
    Response.Redirect (errorURL + "?rc=" + enroll.Rc + "&errorDesc=" +  
enroll.ErrorDesc);  
    return;  
}  
  
// =====  
// =                redirect del client verso URL ACS                =  
// =====  
String enrStatus = enroll.EnrStatus;  
if ("Y".equals(enrStatus)) {  
    Response.getWriter().print(enroll.AcsPage);  
    return;  
}
```

3.5.4.4 .NET MPI Auth

```
// =====
// =          importazione classi di riferimento          =
// =====
using System;
using System.Collections;
using System.Data;
using System.Web;
using System.Text;
using it.netsw.apps.igfs.cg.coms.api.mpi;
// =====
// = impostazione parametri per l'inizializzazione richiesta      =
// = NB: I parametri riportati sono solo a titolo di esempio      =
// =====
String serverURL      = "https://IPGATEWAY/IGFS_CG_SERVICES/services";
int timeout           = 15000;
String tid            = "123456";
String kSig           = "ondkmctaf9/MI3I5AZ4LskbmRiw=";
String shopID        = "5687010820272485455";
String errorURL      = "https://merchant/error.jsp";
String PaRes         = Request["PaRes"];
String md             = Request["MD"];

IgfsCgMpiAuth auth = new IgfsCgMpiAuth();

auth.ServerURL = new Uri(serverURL);
auth.Timeout   = timeout;
auth.Tid       = tid;
auth.KSig      = kSig;
auth.ShopID    = shopID;
auth.PaRes     = PaRes;
auth.Md        = md;

if (!auth.execute()) {
    // =====
    // = redirect del client su pagina di errore definita dall' esercente =
    // =====
    Response.Redirect (errorURL + "?rc=" + auth.Rc + "&errorDesc=" +
auth.ErrorDesc);
    return;
}

// =====
// = Salvo I dati dell'auth per inviarli nella transazione      =
// =====

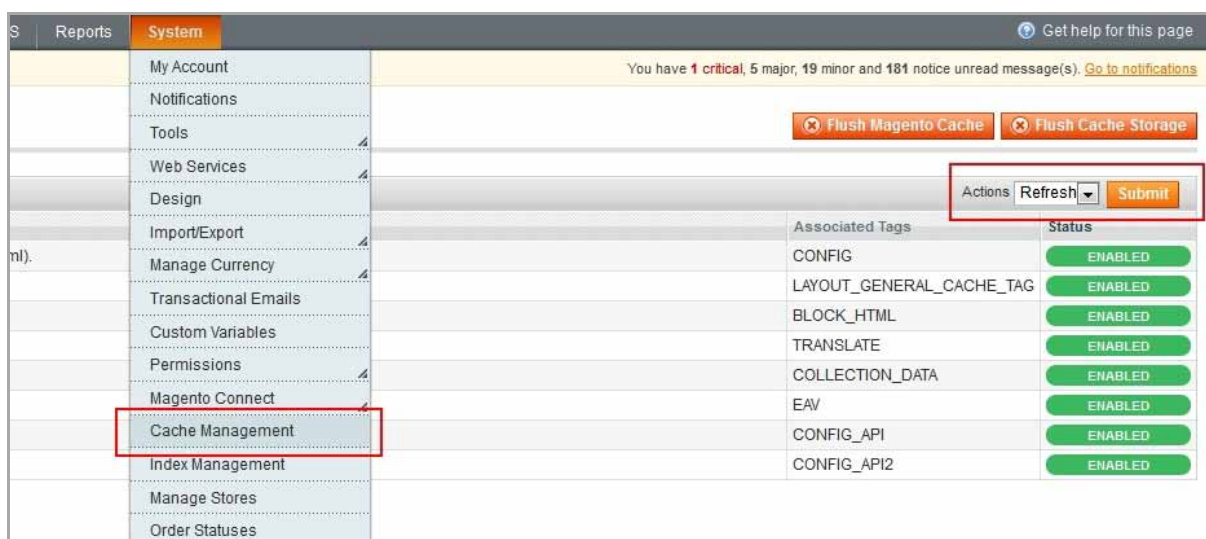
String authStatus = auth.AuthStatus;
String xid        = auth.Xid;
String cavv       = auth.Cavv;
String ecci       = auth.Eci;
```

3.6 Integrazione API con CMS

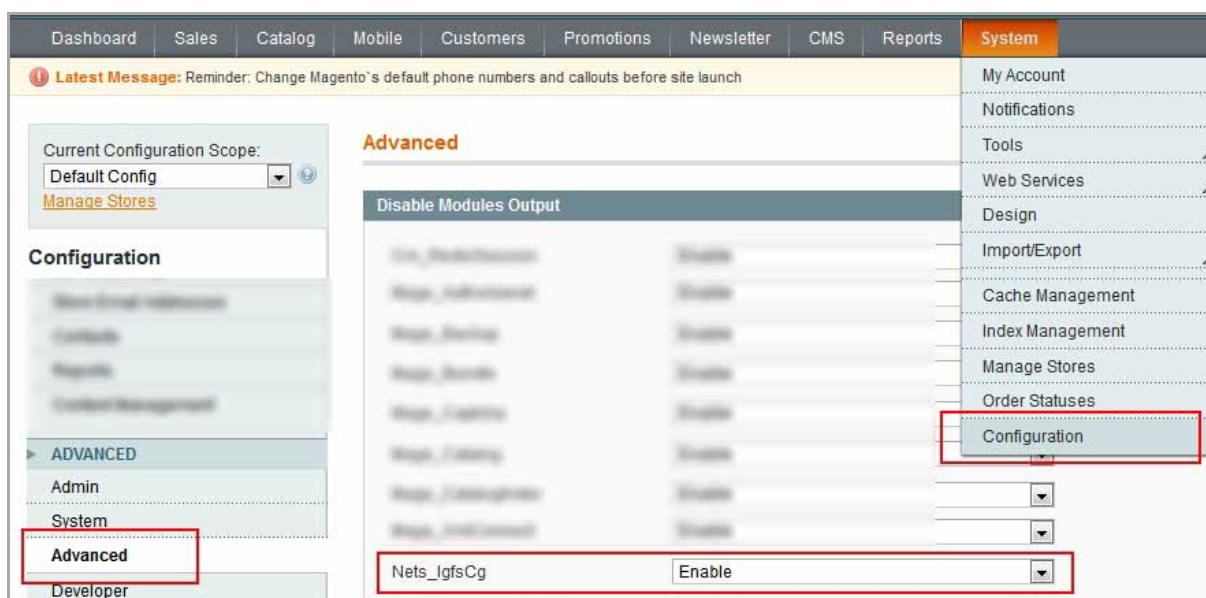
3.6.1 Magento

Estrarre il contenuto di `IGFS_CG_MAGENTO.zip` nella cartella di installazione di Magento. Posizionare le API `IGFS_CG_API` in una cartella a vostro piacere.

Per verificare la corretta installazione del plugin, collegarsi al pannello di amministrazione di Magento e, dopo aver ripulito la Cache come mostrato in figura:



verificare che il modulo `Nets_IgfsCg` sia attivo.



Ora, sempre in configurazione, nel pannello “Payment Methods” dovrebbe essere visibile e configurabile il sistema di pagamento **IGFS Commerce Gateway**.

The screenshot shows the Magento 2 admin interface. The top navigation bar includes Dashboard, Sales, Catalog, Mobile, Customers, Promotions, Newsletter, CMS, Reports, and System. A sidebar on the left contains a menu with categories like SALES, Shipping Settings, and Payment Methods, which is highlighted with a red box. The main content area is titled 'Payment Methods' and features a list of payment methods. 'IGFS Commerce Gateway' is selected and highlighted with a red box. Below the list, the configuration form for 'IGFS Commerce Gateway' is visible, with fields for Title, Enabled, Payment Action, Server URL, Disable Check SSL Certificate, Id Esercente (tid), Chiave (iSig), and Path IGFS_CF_API. A 'Configuration' link in the top right of the list is also highlighted with a red box.

Parametri di configurazione

Tipo[Dimensione]	Property
String	Title Nome del pagamento presentato all'utente in fase di acquisto
YES, NO	Enabled Abilita il metodo di pagamento
AUTH, PURCHASE	Payment Action Tipo di Transazione generata, "Authorize Only" (AUTH) o "Authorize and Capture" (PURCHASE)
URL	ServerURL Indirizzo del server di destinazione della richiesta
YES, NO	Disable Check SSL Certificate Disabilita il controllo del Certificato
String[16]	Tid Codice terminale dell' esercente

String[64]	KSig Chiave per firmare il messaggio
PATH	Path IGFS_CF_API Path installazione di IGFS_CG_API (Se non specificata corrisponde alle lib di magento)

APPENDICE

APPENDICE A: CALCOLO SIGNATURE

L' HMACSHA256 è un tipo di algoritmo con chiave costruito dalla funzione hash SHA-256 e utilizzato come codice HMAC (Hash-based Message Authentication Code).

A titolo di esempio, riportiamo il calcolo dell'algoritmo HMACSHA256 sul campo "signature" attraverso la tecnologia java:

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

...

// =====
// calcolo della signature attraverso l'algoritmo SHA256
//
// key    = chiave segreta
// fields = parametri del messaggio
// =====
public String getSHA256Signature(String key, Object... fields) throws
Exception {
    StringBuilder sb = new StringBuilder();
    for (Object field : fields) {
        if (field != null) {
            sb.append(field.toString());
        }
    }

    byte data[] = sb.toString().getBytes();

    String alg = "HmacSHA256";
    SecretKeySpec sk1 = new SecretKeySpec(key.getBytes(), alg);
    Mac mac = Mac.getInstance(alg);
    mac.init(sk1);
    byte sig[] = mac.doFinal(data);
    return new String(Base64.encode(sig));
}
```

e attraverso la tecnologia .NET

```
using System.Security.Cryptography;
...

static String getSignature(String key, params Object[] fields)
{
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < fields.Length; i++)
    {
        Object field = fields[i];
        if (field != null)
        {
            sb.Append(field.ToString());
        }
    }
    byte[] keyByte = Encoding.UTF8.GetBytes(key);
    HMACSHA256 hmacsha256 = new HMACSHA256(keyByte);
    byte[] data = Encoding.UTF8.GetBytes(sb.ToString());
    byte[] mac = hmacsha256.ComputeHash(data);
    return System.Convert.ToBase64String(mac);
}
```

NB: La classe Base64 implementa l'encoding in Base64 dei byte componenti la signature.

APPENDICE B: FILE BATCH

Di seguito viene illustrata la struttura dei files batch per operare in **IGFS®** opzione **BUY NOW**.

Ogni transazione è rappresentata da un record del file, i campi sono separati dalla virgola “,”.

Nel caso in cui un campo sia opzionale e non inserito sul file deve essere inserito il campo vuoto.

La struttura inviata in input presenta già i campi vuoti che verranno valorizzati in fase di output.

FILE DI INPUT

La struttura di INPUT segue lo schema:

TRTYPE,TID,AMOUNT,CURRENCY,REFTRANID,SHOPID,PAYINSTRTOKEN,,,,,
SHOPUSERREF,ADDINFO1,ADDINFO2,ADDINFO3,ADDINFO4,ADDINFO5, TOPUPID,SPLITTRAN,

CAMPO	DESCRIZIONE	M/O	NOTE
TRTYPE	Tipo di operazione da effettuare	M	D=DEBIT (CONFERMA) V=VOID (STORNO) C=CREDIT (CREDITO) A=AUTH (AUTORIZZAZIONE) P=PURCHASE (ADDEBITO)
TID	Codice terminale	M	Codice terminale utilizzato per la transazione.
AMOUNT	Importo della transazione con il carattere "punto" (.) come separatore dei decimali	M	ES: 1 Euro => "1.00". Si utilizza il numero di decimali previsti dagli standard per la relativa valuta.
CURRENCY	Codice numerico identificativo della valuta	M	ES: 978 per Euro.
REFTRANID	Riferimento TranID	O	Per TRTYPE "D" o "V" il REFTRANID atteso è quello ottenuto in fase di autorizzazione attraverso la proprietà TRANID (Nel caso di TRTYPE "D" relativo ad operazioni di TOPUP tale valore non deve essere impostato). Per TRTYPE "C" deve essere inserito il campo TRANID ottenuto sulla conferma contabile. Per TRTYPE "A" o "P" non deve essere impostato.
SHOPID	Chiave esterna identificante l'operazione	M	
PAYINSTRTOKEN	Token sostitutivo dello strumento di pagamento	O	Per TRTYPE "A" o "P" deve essere impostato con un TOKEN sostitutivo dello strumento di pagamento valido.
SHOPUSERREF	Identificativo cliente	O	ES: email.
ADDINFO1	Campo a disposizione dell'esercente	O	
ADDINFO2	Campo a disposizione dell'esercente	O	
ADDINFO3	Campo a disposizione dell'esercente	O	
ADDINFO4	Campo a disposizione dell'esercente	O	
ADDINFO5	Campo a disposizione dell'esercente	O	
TOPUPID	ID riferimento TopUp	O	Per TRTYPE "A" può essere impostato per identificare un'operazione di TOPUP. Per TRTYPE "D" deve essere impostato col valore utilizzato per le transazioni di autorizzazione.
SPLITTRAN	Flag di conferma parziale	O	Y/N

FILE DI OUTPUT

La struttura di OUTPUT segue lo schema:

TRTYPE,TID,AMOUNT,CURRENCY,REFTRANID,SHOPID,PAYINSTRTOKEN,RC,DESC,TRANID,
SHOPUSERREF,ADDINFO1,ADDINFO2,ADDINFO3,ADDINFO4,ADDINFO5,TOPUPID,SPLITTRAN,

CAMPO	DESCRIZIONE	M/O	NOTE
TRTYPE	Tipo di operazione da effettuare	M	REPLICATO DA FILE DI INPUT
TID	Codice terminale	M	REPLICATO DA FILE DI INPUT
AMOUNT	Importo della transazione con il carattere "punto" (.) come separatore dei decimali	M	REPLICATO DA FILE DI INPUT
CURRENCY	Codice numerico identificativo della valuta	M	REPLICATO DA FILE DI INPUT
REFTRANID	Riferimento TranID	O	REPLICATO DA FILE DI INPUT
SHOPID	Chiave esterna identificante l'operazione	M	REPLICATO DA FILE DI INPUT
PAYINSTRTOKEN	Token sostitutivo dello strumento di pagamento	O	REPLICATO DA FILE DI INPUT
RC	Codice di risposta	M	Vedi APPENDICE C: CODICI RITORNO.
DESC	Descrizione RC	M	Descrizione del codice di risposta, in caso di errore fornisce informazioni aggiuntive.
TRANID	TranID	M	Fornisce un riferimento alla transazione di effettuata. Il dato è presente solo in caso di azione conclusa correttamente.
SHOPUSERREF	Identificativo cliente	O	REPLICATO DA FILE DI INPUT
ADDINFO1	Campo a disposizione dell'esercente	O	REPLICATO DA FILE DI INPUT
ADDINFO2	Campo a disposizione dell'esercente	O	REPLICATO DA FILE DI INPUT
ADDINFO3	Campo a disposizione dell'esercente	O	REPLICATO DA FILE DI INPUT
ADDINFO4	Campo a disposizione dell'esercente	O	REPLICATO DA FILE DI INPUT
ADDINFO5	Campo a disposizione dell'esercente	O	REPLICATO DA FILE DI INPUT
TOPUPID	ID riferimento TopUp	O	REPLICATO DA FILE DI INPUT
SPLITTRAN	Flag di conferma parziale	O	REPLICATO DA FILE DI INPUT

APPENDICE C: CODICI RITORNO

CODICE	DESCRIZIONE
IGFS_000	TRANSAZIONE OK
IGFS_001	DESTINATARIO SCONOSCIUTO
IGFS_00155	ID BATCH NON VALIDO
IGFS_00156	ID BATCH NON UNIVOCO
IGFS_00157	STRUMENTO PAGAMENTO NON VALIDO
IGFS_00158	NUMERO CARTA NON NUMERICO
IGFS_00159	NUMERO CARTA NON PRESENTE
IGFS_002	CARTA SCADUTA
IGFS_00260	L'IMPORTO DEL CREDITO SUPERA L'IMPORTO DEL MOVIMENTO
IGFS_00261	L'IMPORTO DEL MOVIMENTO SUPERA L'IMPORTO DELL' AUTORIZZAZIONE
IGFS_003	CARTA ERRATA
IGFS_004	CARTA IN BLACK LIST
IGFS_00452	CODICE TERMINALE NON PRESENTE
IGFS_00456	CODICE TERMINALE ERRATO
IGFS_005	ERRORE DI FORMATO
IGFS_006	ERRORE FILE SYSTEM
IGFS_007	ERRORE DI COMUNICAZIONE
IGFS_00701	BATCH ID NON PROCESSATO
IGFS_00704	BATCH ID NON NUMERICO
IGFS_00705	BATCH ID NON PRESENTE
IGFS_008	AUTORIZZAZIONE NEGATA
IGFS_009	RITIRARE CARTA
IGFS_00950	DIRECTORY BATCH UPLOAD NON PRESENTE
IGFS_00951	DIRECTORY BATCH DOWNLOAD NON PRESENTE
IGFS_00952	NOME DIRECTORY ARCHIVIAZIONE BATCH NON PRESENTE
IGFS_010	MERCHANT NON ABILITATO
IGFS_01000	TRANSAZIONE NEGATA DAL SISTEMA ANTIFRODE
IGFS_011	CONTATTARE ACQUIRER
IGFS_014	MERCHANT NON CONVENZIONATO
IGFS_015	CARTA NON GESTITA
IGFS_016	CARTA IN RANGE NEGATIVO O STRANIERA
IGFS_018	CARTA INESISTENTE
IGFS_020	CARTA INVALIDA
IGFS_021	CODICE MERCHANT ERRATO
IGFS_029	DATA SCADENZA ERRATA
IGFS_030	FONDI INSUFFICIENTI
IGFS_032	IMPORTO NON VALIDO

CODICE	DESCRIZIONE
IGFS_033	TRANSAZIONE ORIGINALE NON TROVATA
IGFS_083	ERRORE CIFRATURA TRANSAZIONE
IGFS_085	CODICE DIVISA ERRATO
IGFS_086	MALFUNZIONAMENTO SISTEMA
IGFS_087	ACQUIRER NON RAGGIUNGIBILE
IGFS_088	MANCATA RISPOSTA DA ACQUIRER
IGFS_091	MALFUNZIONAMENTO SISTEMA ACQUIRER
IGFS_092	TRANSAZIONE SCONOSCIUTA
IGFS_093	CONFERMA GIA' PRESENTE
IGFS_095	STORNO PER NOTIFICA INESISTENTE
IGFS_096	STORNO PER AUTORIZZAZIONE INESISTENTE
IGFS_097	CONFERMA PER AUTORIZZAZIONE INESISTENTE
IGFS_098	IMPORTO SUPERIORE AD IMPORTO AUTORIZZATO
IGFS_10000	CARATTERI NON VALIDI
IGFS_101	MAC ERRATO
IGFS_102	SOSPETTA FRODE
IGFS_104	CARTA SOGGETTA A RESTRIZIONI
IGFS_107	CONTATTARE ISSUER
IGFS_108	CONTATTARE ISSUER: CASO SPECIALE
IGFS_112	INSERIRE PIN
IGFS_115	FUNZIONE NON SUPPORTATA SU CARTA
IGFS_117	PIN ERRATO
IGFS_118	CONTO NON TROVATO O NON ABILITATO
IGFS_119	OPERAZIONE NON PERMESSA AL TITOLARE
IGFS_121	SUPERATO LIMITE IMPORTO
IGFS_122	ERRORE SICUREZZA
IGFS_123	SUPERATO LIMITE FREQUENZA
IGFS_125	CARTA NON ATTIVA
IGFS_129	SOSPETTA FRODE SU CARTA
IGFS_160	CARTA PERSA
IGFS_164	DATA ANTEC. A BLOCCO CARTA
IGFS_180	DATI ERRATI
IGFS_181	DATI SENSIBILI ERRATI
IGFS_1921	3DS: IMPOSSIBILE AUTENTICARE CARTA (PARES=U)
IGFS_1922	3DS: AUTENTICAZIONE NON AVVENUTA (PARES=N)
IGFS_1923	3DS: IMPOSSIBILE VERIFICARE ISCRIZIONE CARTA (VERES=U)
IGFS_20000	DATI MANCANTI
IGFS_20001	CODICE OPERAZIONE NON VALIDO
IGFS_20002	SESSIONE SCADUTA
IGFS_20003	COOKIE NON ABILITATI

CODICE	DESCRIZIONE
IGFS_20007	STATO ORDINE NON VALIDO
IGFS_20010	URL INVIO RISPOSTA NON VALIDO
IGFS_20011	URL INVIO ERRORE NON VALIDO
IGFS_20012	SHOPID NON VALIDO
IGFS_20013	CODICE LINGUA NON VALIDO
IGFS_20014	CAMPO AGGIUNTIVO NON VALIDO
IGFS_20018	CVV2 NON VALIDO
IGFS_20019	SHOPID NON VALIDO
IGFS_20020	CAMPO ADDIZIONALE NON VALIDO
IGFS_20021	CAMPO API VERSION NON VALIDO
IGFS_20022	CAMPO SIGNATURE NON VALIDO
IGFS_20023	CAMPO PAYMENT ID NON VALIDO
IGFS_20024	CODICE AUTORIZZAZIONE MANCANTE
IGFS_20025	CAMPO REFERENCE DATA NON VALIDO
IGFS_20026	SHOP ID DUPLICATO
IGFS_20027	RICHIESTA BATCH NON VALIDA
IGFS_20028	DATI BATCH MANCANTI
IGFS_20029	DATI BATCH NON VALIDI
IGFS_20030	DIRECTORY DATI BATCH NON VALIDA
IGFS_20031	DATI BATCH DUPLICATI
IGFS_20032	NOME BATCH FILE NON VALIDO
IGFS_20033	DATI BATCH NON TROVATI
IGFS_20034	BATCH SHOPID NON VALIDO
IGFS_20035	ID ORDINE NON VALIDO
IGFS_20036	PAN NON VALIDO
IGFS_20037	CVV2 NON VALIDO
IGFS_20038	DATA SCADENZA ERRATA
IGFS_20044	DESCRIZIONE PAGAMENTO NON VALIDA
IGFS_20090	TRANSAZIONE CANCELLATA DALL'UTENTE
IGFS_20100	ERRORE NOTIFICA MERCHANT
IGFS_800	TERMINALE NON ABILITATO
IGFS_801	BANCA SELEZIONATA ERRATA
IGFS_802	TENTATIVI PIN ESAURITI
IGFS_803	CODICE TERMINALE ERRATO
IGFS_804	CHIAVE DISALLINEATA
IGFS_805	ERRORE CIFRATURA
IGFS_807	TERMINALE CHIUSO
IGFS_808	TERMINALE NON CHIUSO
IGFS_809	ERRORE SEQUENZA
IGFS_810	TERMINALE NON RICONOSCIUTO

CODICE	DESCRIZIONE
IGFS_811	TERMINALE BLOCCATO
IGFS_812	TERMINALE CHIUSO FORZATAMENTE
IGFS_813	OPERAZIONE NON PERMESSA
IGFS_814	TRANSAZIONE IN CORSO
IGFS_815	CARTA BLOCCATA
IGFS_90000	DATABASE ERROR
IGFS_90005	TIMESTAMP ERRATO
IGFS_902	TRANSAZIONE NON VALIDA
IGFS_903	REINVIARE TRANSAZIONE
IGFS_907	EMITTENTE NON ADERENTE
IGFS_908	DESTINAZIONE NON TROVATA
IGFS_909	ERRORE DI SISTEMA
IGFS_910	SISTEMA ISSUER NON ATTIVO
IGFS_911	TIME OUT
IGFS_912	ISSUER NON RAGGIUNGIBILE
IGFS_913	TRANSAZIONE DUPLICATA
IGFS_990	STRUMENTO PAGAMENTO NON ATTIVO