3D Pay

Entegrasyon Dokümanı

İnnova Bilişim Çözümleri A.Ş.

1. Giriş ve Genel Bakış

Bu doküman, Ziraat Bankası'nın İNNOVA altyapısı ile sağladığı yeni nesil sanal POS çözümü olan **3D Pay** entegrasyonuna yönelik teknik bilgileri içermektedir.

Klasik 3D Secure yönteminde iki aşamalı yürütülen kart doğrulama ve tahsilat işlemleri, bu modelde **tek adımda ve arka planda otomatik** olarak gerçekleştirilmektedir.

3D Pay'in Avantajları

- **Tek Adım İşlem:** Doğrulama ve tahsilat birleşik
- **Gelişmiş Güvenlik:** PCI DSS uyumlu, kart bilgileri merchant'ta tutulmaz
- **Kolay Entegrasyon:** Manuel tahsilat çağrısı gerektirmez
- **İyileştirilmiş UX:** Kullanıcı deneyimi optimize edilmiş

2. 3D Pay Nedir?

3D Pay, kartlı ödeme işlemlerinde kart güvenliği sağlayan 3D Secure teknolojisinin **doğrulama adımı ile tahsilat işlemini tek bir işlem içinde birleştiren** yeni nesil sanal POS modelidir.

Bu modelde, kart sahibinin doğrulama işlemi (OTP - SMS şifresi veya mobil onay) başarılı şekilde tamamlandığında, ödeme tutarı **sistem tarafından arka planda otomatik olarak tahsil edilir**.

Klasik 3D Secure vs 3D Pay

Klasik 3D Secure	3D Pay
İki aşamalı işlem	**Tek aşamalı işlem**
Doğrulama sonrası manuel VPOS API çağrısı	**Otomatik tahsilat**
Yüksek entegrasyon maliyeti	**Düşük entegrasyon maliyeti**
Merchant tahsilat sorumluluğu	**Gateway tahsilat sorumluluğu**

3. Temel Tanımlar

Terim	Açıklama
3D Pay	Kart doğrulama ve tahsilat işleminin, kullanıcının doğrulaması sonrasında banka altyapısı üzerinden otomatik olarak gerçekleştirildiği ödeme modelidir.
StoreType	3D Pay modelinin kullanılacağını belirtir. Değer: `3d_pay`
TransactionType	İşlem tipini belirtir. Desteklenen değerler: `Sale` (tahsilat), `Auth` (ön provizyon)
HashData	SHA-512 algoritması ile hesaplanmış güvenlik imzasıdır. İsteğin bütünlüğünü ve kaynağını doğrulamak için kullanılır.
okUrl / failUrl	İşlemin başarılı ya da başarısız olmasına göre kullanıcının yönlendirileceği URL adresleridir.
ProcReturnCode	İşlem sonuç kodudur. `0000` başarılı işlemi gösterir.
PCI DSS	Payment Card Industry Data Security Standard - Kart verisi güvenlik standardı.

4. Erişim Bilgileri

Servis	Test Ortamı	Prod Ortamı
3D Pay	https://sanalpos.innova.com.tr/ZIRAATBANK/VposWeb/v3/VposThreeDPay.aspx	https://sanalpos.ziraatbank.com.tr/v4/v3/VposThreeDPay.aspx
Test Form URL	https://sanalpos.innova.com.tr/ziraatbankasi/3dpaytest.html	

Erişim Bilgileriniz

Entegrasyon için aşağıdaki bilgilere ihtiyacınız olacak:

- **MerchantId:** Üye işyeri numaranız (15 karakter)
- **MerchantPassword:** Hash hesaplama için kullanılacak şifreniz

5. 3D Pay İşlem Akışı

3D Pay işlem akışı, kullanıcıdan alınan kart bilgilerinin arka planda doğrulanması ve tahsilatın otomatik olarak gerçekleştirilmesi sürecini kapsar.

İşlem Adımları

1.Kart Bilgisi Girişi: Kullanıcı, merchant arayüzünde kart bilgilerini girer.
2.İstek Hazırlama: Merchant, istek parametrelerini hazırlar ve HashData hesaplar (SHA-512).
3.3D Pay Servisine POST: Kart bilgileri PCI DSS uyumlu şekilde 3D Pay servisine iletilir.
4.3D Secure Yönlendirme: Sistem, kullanıcıyı banka 3D Secure OTP ekranına yönlendirir.
5.OTP Doğrulama: Kullanıcı, bankadan gelen OTP'yi girerek işlemi onaylar.
6.Otomatik Tahsilat: Doğrulama başarılıysa, tahsilat otomatik olarak arka planda gerçekleşir.
7.Sonuç Bildirimi: İşlem sonucu okUrl/failUrl'e callbacktype (post veya get) parametresine göre kullanıcı dönüşü yapılır.

6. İstek Parametreleri

3D Pay servisine POST isteği gönderirken aşağıdaki parametreler kullanılır.

Zorunlu Parametreler

Parametre Adı	Zorunlu	Veri Tipi / Format	Açıklama	Örnek Değer
MerchantId	✓	Alfanumerik 15 karakter	Üye işyeri numarası. Banka tarafından sağlanır.	000000000281567
OrderId	✓	Alfanumerik Max 64 karakter	Benzersiz sipariş numarası. Her işlem için farklı olmalıdır.	ORD-2025-00123
StoreType	✓	String Sabit değer	3D Pay işlemleri için sabit değer	3d_pay
HashData	✓	String Base64 (SHA-512)	Güvenlik hash değeri. Bölüm 4'e bakınız.	gH7k9mN2pQ4...
TransactionType	✓	String 4-6 karakter	İşlem tipi: Sale: Satış Auth: Ön Provizyon Cancel: İptal Refund: İade	Sale
CurrencyAmount	✓	Decimal Nokta ayraçlı 2 ondalık	İşlem tutarı. Kuruş dahil, nokta ile ayrılmış.	10.99
CurrencyCode	✓	Numeric 3 hane	Para birimi kodu 949: TL 840: USD 978: EUR	949
Pan	✓	Numeric 16-19 hane	Kredi kartı numarası. Boşluk ve tire olmadan.	4938410158942300
Cvv	✓	Numeric 3-4 hane	Kart güvenlik kodu (CVV2/CVC2)	123

			Amex için 4 hane	
Expiry	✓	Numeric 6 hane YYYYMM	Kartın son kullanma tarihi Format: YYYYMM	202512
ClientIp	✓	String IPv4	İşlemi yapan müşterinin IP adresi	192.168.1.100
okUrl	✓	String URL	Başarılı işlem sonrası yönlendirme adresi	https://merchant.com/success
failUrl	✓	String URL	Başarısız işlem sonrası yönlendirme adresi	https://merchant.com/fail
NumberOfInstallments		Numeric 1-2 hane	Taksit sayısı. 1 veya boş: Peşin 2-12: Taksitli	3
MerchantType		Numeric 1 hane	Bayi yapısı: 1: Ana Bayi 2: Alt Bayi	1
TransactionDeviceSource		Numeric 1 hane	İşlem kaynağı: 0: E-Commerce (Web) 1: Mail Order 3: Mobile	0
BrandName		Numeric 3 hane	Kart markası: 100: Visa 200: MasterCard 300: Troy 400: Amex	100
OrderDescription		Alfanumerik Max 2000 kar.	Sipariş açıklaması/ detayı	Ürün satışı - Ödeme
CardHoldersName		Alfanumerik Max 100 karakter	Kart üzerindeki isim	EMINE TOPAL
Extract		Alfanumerik Max 40 karakter	Ekstrde görünecek ek bilgi	090020304

			(abone no, vb.)	
Location		Numeric 1 hane	İşlem yeri: 1: İnternet 2: Telefon 3: Mobil 4: ERP	1
RND	✓		32 karakterlik benzersiz GUID değeri (tiresiz, hexadecimal). Replay attack önleme için kullanılır. Her işlem için farklı olmalıdır.	
CallbackType		**GET/POST**	Bu alan zorunlu değildir. Varsayılan (default) olarak **GET** metodu kullanılmaktadır. Alan kullanılarak işlem sonucunun kullanıcıya hangi HTTP metodu (**GET / POST**) ile iletileceği belirlenebilir.	

KRİTİK UYARI: İstekte asla `password` parametresi göndermeyin! Bu parametre gönderilirse istek otomatik olarak reddedilir. Sadece `HashData` kullanılmalıdır.

7. Yanıt Parametreleri

İşlem tamamlandıktan sonra **SuccessUrl** veya **FailUrl** adresinize POST edilen parametreler:

Parametre	Açıklama
Oid	Sipariş numarası (gönderdiğiniz OrderId)
AuthCode	Banka onay kodu (başarısız işlemlerde boş)
ResultMessage	İşlem sonuç mesajı
TransId	Banka işlem ID'si
HostRefNum	Banka referans numarası
ProcReturnCode	İşlem sonuç kodu (00 = başarılı)
Response	İşlem sonucu (Approved / Error)
TrxDate	İşlem tarihi (AYGGSSMM formatı)
SuccessUrl	Gönderdiğiniz başarı URL'i
FailUrl	Gönderdiğiniz hata URL'i
MdStatus	3D Secure doğrulama durumu
hashParams	Hash hesaplamasında kullanılan parametre sırası
hash	Yanıt bütünlüğü doğrulama hash'i
hashItems	hashParams içinde belirtilen alanların değerlerinin, belirtilen sıraya göre ve araya ayırıcı karakter konulmadan birleştirilmiş halidir.

NOT: İşlem sonucunu kontrol etmek için **ProcReturnCode** parametresine bakmanız yeterlidir. **`0000`** değeri başarılı işlemi gösterir.

7.1 Yanıt Hash Doğrulaması

3D Pay Servisi, gönderdiği yanıtın bütünlüğünü ve gerçekten 3D Pay Servisinden geldiğini kanıtlamak amacıyla aşağıdaki üç alanı gönderir:

- hashParams
- hashItems
- hash

Hash doğrulaması zorunlu değildir; ancak güvenlik açısından yapılması önerilir.

7.1.1 hashParams

Hash hesaplamasına dahil edilen parametrelerin sırasını belirtir.

Örnek:

hashParams = Oid+ErrMsg+ProcReturnCode+Response+SuccessUrl+FailUrl

Bu ifade, hash hesabında kullanılacak alanların sırasını gösterir.

7.1.2 hashItems

hashParams içinde belirtilen alanların değerlerinin, belirtilen sıraya göre ve araya ayırıcı karakter konulmadan birleştirilmiş halidir.

Örnek Yanıt:

Parametre	Değer
Oid	ORD-202602131322442923
ErrMsg	HashData geçersiz
ProcReturnCode	4096
Response	Error
SuccessUrl	https://www.google.com/success
FailUrl	https://www.google.com/fail

hashParams sırasına göre birleştirme:

hashItems =
ORD-202602131322442923
HashData geçersiz
4096
Error
https://www.google.com/success
https://www.google.com/fail

Sonuç:

ORD-202602131322442923HashData geçersiz4096Errorhttps://www.google.com/successhttps://www.google.com/fail

3D Pay Servisi bu değeri hazır olarak gönderir.

7.1.3 hash

3D Pay Servisi, hashItems değerinin sonuna MerchantPassword ekleyerek aşağıdaki işlemi uygular:

hashItems + MerchantPassword
↓
SHA-512
↓
Base64 Encode
↓
hash

Örnek:

hashItems = ORD-202602131322442923HashData geçersiz4096Error
https://www.google.com/successhttps://www.google.com/fail

MerchantPassword = ********

SHA-512 + Base64 sonucu:

FPY2VsbAIL2jXesHlryNDUOHEVnggs9gKZirenR2nHwnpLgCGih6+
9yJesD/clCEKQJZd+jAxxIm2lWqU4NChw==

7.1.4 Doğrulama Adımları

Üye işyeri tarafında yapılması gereken işlem:

1. 3D Pay Servisinden gelen **hashItems** değeri alınır.
2. Sonuna kendi **MerchantPassword** değeri eklenir.
3. **SHA-512** algoritması ile hash hesaplanır.
4. Sonuç **Base64** encode edilir.
5. Hesaplanan değer, 3D Pay Servisinden gelen **hash** değeri ile karşılaştırılır.

Eşleşirse: Yanıt 3D Pay Servisinden gelmiştir ve bütünlüğü korunmuştur. ✓

Eşleşmezse: Yanıt geçersiz kabul edilmelidir. ✗

Zorunluluk Durumu

Hash doğrulaması teknik olarak zorunlu değildir. Ancak yapılmaması durumunda, üçüncü kişiler tarafından SuccessUrl adresine sahte işlem bildirimi gönderilmesi riski oluşabilir.

Bu nedenle güvenlik açısından hash doğrulaması yapılması önerilir.

8. Hash Güvenlik Mekanizması

3D Pay entegrasyonunda, isteğin güvenliğini sağlamak için **SHA-512 hash algoritması** kullanılır. Bu mekanizma, isteğin kaynağını doğrular ve verinin değiştirilmediğini garanti eder.

Hash Hesaplama

Parametreler (Sıralı)

MerchantId + OrderId + CurrencyCode + CurrencyAmount(kuruş) + okUrl + failUrl + Rnd + MerchantPassword

Önemli Notlar

- **CurrencyAmount değeri 100 ile çarpılır ve ondalık kısmı atılır**
  Örnek: 10.99 TL → 1099
- Parametreler arasında boşluk veya ayırıcı karakter yoktur
- MerchantPassword size özel olarak verilen şifredir
- **Hash hesaplandıktan sonra HashData parametresi olarak gönderilir**

Rnd Nedir?

- 32 karakterlik benzersiz GUID değeri (tiresiz, hexadecimal)
- Her işlem için FARKLI bir değer olmalıdır
- Format: ^[a-fA-F0-9]{32}$

Neden Kullanılır?

- Replay Attack (tekrar saldırısı) önleme
- Aynı Rnd değeri ile ikinci istek otomatik olarak REDDEDİLİR
- İşlem güvenliğini artırır

9.Kod Örnekleri

C# Örneği

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

// Rnd Değeri Oluştur
string rnd = Guid.NewGuid().ToString("N");

// SHA-512 Hash Fonksiyonu
private static string HashSHA512(string hashString) {
    SHA512 sHA = new SHA512CryptoServiceProvider();
    byte[] bytes = Encoding.ASCII.GetBytes(hashString);
    byte[] inArray = sHA.ComputeHash(bytes);
    return Convert.ToBase64String(inArray);
}

// Hash String Oluşturma
StringBuilder hash = new StringBuilder();
hash.Append(merchantId);
hash.Append(orderId);
hash.Append(currencyCode);
hash.Append((long)Math.Truncate(currencyAmount * 100m));
hash.Append(okUrl);
hash.Append(failUrl);
hash.Append(rnd);
hash.Append(merchantPassword);

string hashData = HashSHA512(hash.ToString());

vposRequest.Rnd = rnd;
vposRequest.HashData = hashData;
```

PHP Örneği

```
<?php
use Ramsey\Uuid\Uuid;

$rnd = str_replace('-', '', Uuid::uuid4()->toString());

function hashSHA512($hashString) {
    return base64_encode(hash('sha512', $hashString, true));
}

$hashString = $merchantId . $orderId . $currencyCode .
              (int)($currencyAmount * 100) . $okUrl . $failUrl .
              $rnd . $merchantPassword;

$hashData = hashSHA512($hashString);
$vposRequest['Rnd'] = $rnd;
$vposRequest['HashData'] = $hashData;?>
```

Python Örneği

```
import hashlib
import base64
import uuid

rnd = uuid.uuid4().hex

def hash_sha512(hash_string):
    hash_bytes = hashlib.sha512(hash_string.encode('ascii')).digest()
    return base64.b64encode(hash_bytes).decode('ascii')

hash_string = (merchant_id + order_id + currency_code +
               str(int(currency_amount * 100)) + ok_url + fail_url +
               rnd + merchant_password)

hash_data = hash_sha512(hash_string)

vpos_request['Rnd'] = rnd
vpos_request['HashData'] = hash_data
```

Java Örneği

```
import java.security.MessageDigest;
import java.util.Base64;
import java.util.UUID;

String rnd = UUID.randomUUID().toString().replace("-", "");

public static String hashSHA512(String hashString) throws Exception {
    MessageDigest md = MessageDigest.getInstance("SHA-512");
    byte[] bytes = hashString.getBytes("ASCII");
    byte[] digest = md.digest(bytes);
    return Base64.getEncoder().encodeToString(digest);
}

StringBuilder hash = new StringBuilder();
hash.append(merchantId).append(orderId).append(currencyCode)
    .append((long)(currencyAmount * 100)).append(okUrl).append(failUrl)
    .append(rnd).append(merchantPassword);

String hashData = hashSHA512(hash.toString());
vposRequest.setRnd(rnd);
vposRequest.setHashData(hashData);
```

10. Test Aracı

10.1 Amaç

3D Pay entegrasyonunu doğrulamak ve canlı ortama geçmeden önce uçtan uca test gerçekleştirebilmek amacıyla, tarayıcı üzerinden çalışan hazır bir HTML test aracı sağlanmaktadır.

Bu araç sayesinde merchant, herhangi bir yazılım geliştirmeye ihtiyaç duymadan:

- Parametre bilgilerini girerek
- Hash değerini oluşturarak
- Test ortamına doğrudan istek göndererek

entegrasyonunu doğrulayabilir.

Test aracı, **3DPay_Test_Tool.html** dosyasının tarayıcıda açılmış halidir.

Ayrıca test ortamı için hazır form aşağıdaki adresten erişilebilir:

Test Form URL:
https://sanalpos.innova.com.tr/ziraatbankasi/3dpaytest.html

⚠ Bu adres yalnızca test ortamı için kullanılmalıdır.

10.2 Dosya Adı

3DPay_Test_Tool.html

Bu dosya doküman ile birlikte sağlanır.
Dosya mevcut değilse, dokümanın ekinde verilen kaynak kod kullanılarak manuel olarak oluşturulabilir.

10.3 Özellikler

Test aracı aşağıdaki fonksiyonları sağlar:

- Otomatik **Rnd** üretimi (32 karakter hexadecimal)
- Otomatik **OrderId** oluşturma (zaman damgalı, benzersiz)
- **HashData** değerinin otomatik hesaplanması
- Test kartı bilgilerinin hazır gelmesi
- Test ortamına doğrudan POST gönderimi
- Kurulum gerektirmeden tarayıcıda çalışması

10.4 Nasıl Kullanılır?

1) Dosyayı açın

`3DPay_Test_Tool.html` dosyasına çift tıklayın. Test ekranı tarayıcıda açılır.

2) MerchantId girin

MerchantId alanına size bankadan verilen üye işyeri numarasını yazın.

3) Otomatik alanları kontrol edin

- **OrderId** sistem tarafından otomatik oluşturulur ve gerekirse değiştirilebilir.
- **Rnd** değeri otomatik üretilir ve her işlem için farklıdır.

4) Hash oluşturun

"Hash Hesapla & İmzala" butonuna tıklayın.
Açılan pencerede **MerchantPassword** girilir.
Şifre hiçbir zaman form alanlarına veya isteğe eklenmez.

5) İsteği gönderin

"Ödemeyi Başlat (Gönder)" butonuna basarak test ortamına ödeme isteği gönderilir.

3D Pay Test Formu

3D PAY Entegrasyon Dokümanına buradan erişebilirsiniz

ÖDEME BİLGİLERİ

MerchantId

000000000281567

OrderId

ORD-202602131419275304 Oluştur

StoreType

3d_pay

Para Birimi (949=TL)

949

Tutar (1.00)

1

GÜVENLİK & HASH

Rnd (Random Değer)

a1b6c856d2cc202cfa3598078b7e7943 Yenile

HashData

Hesapla butonuna basın

Hash Hesapla & İmzala

DÖNÜŞ URL'LERİ

SuccessUrl

https://www.google.com/success

FailUrl

https://www.google.com/fail

KART BİLGİLERİ (OPSİYONEL)

Pan (Kart No)

5549601963997012

Expiry (YYYYAA)

202909

Cvv

259

CALLBACKTYPE (DEFAULT GET)

CallbackType

GET

Ödemeyi Başlat (Gönder)

10.5 Test Kartı Bilgileri

Alan	Değer
Kart No	4546720000621074
CVV	490
Son Kullanma	202605

Bu kart yalnızca test amaçlıdır, gerçek para çekmez.

10.6 Güvenlik ve Çalışma Prensipleri

- MerchantPassword dosyada saklanmaz ve forma eklenmez.
- Hash hesaplama tamamen tarayıcıda yapılır.
- Şifre sunucuya gönderilmez.
- Rnd değeri her işlem için farklıdır ve tekrar kullanım engellenir.
- Test aracı varsayılan olarak **test ortamına** istek gönderir.
- Canlı ortamda kullanılacaksa, HTML dosyasındaki `form action` adresi banka tarafından verilen prod URL ile değiştirilmelidir.

10.7 HTML Dosyası Oluşturma

Dosya mevcut değilse:

1. Kaynak kodu kopyalayın
2. Notepad açın
3. Yapıştırın
4. **Farklı Kaydet → 3DPay_Test_Tool.html**
5. Dosyaya çift tıklayın

Dosya uzantısı `.html` olmalıdır.

KAYNAK KOD

```
<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3D Pay Test Paneli</title>
    <style>
        :root {
            --primary-color: #2c3e50;
            --accent-color: #3498db;
            --bg-color: #f4f7f6;
            --card-bg: #ffffff;
            --border-color: #e0e0e0;
            --text-color: #333;
        }
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

```
    background-color: var(--bg-color);
    color: var(--text-color);
    display: flex;
    justify-content: center;
    align-items: flex-start;
    min-height: 100vh;
    margin: 0;
    padding: 20px;
}
.container {
    background-color: var(--card-bg);
    padding: 30px;
    border-radius: 12px;
    box-shadow: 0 10px 25px rgba(0,0,0,0.05);
    width: 100%;
    max-width: 500px;
}
h2 {
    text-align: center;
    color: var(--primary-color);
    margin-bottom: 25px;
    font-weight: 600;
    border-bottom: 2px solid var(--bg-color);
    padding-bottom: 15px;
}
.form-group { margin-bottom: 15px; }
label {
    display: block;
    margin-bottom: 6px;
    font-weight: 500;
    font-size: 0.9rem;
    color: #555;
}
.input-group { display: flex; gap: 10px; }
input[type="text"] {
    width: 100%;
    padding: 10px 12px;
    border: 1px solid var(--border-color);
    border-radius: 6px;
    font-size: 14px;
    transition: border-color 0.3s ease;
    box-sizing: border-box;
}
input[type="text"]:focus {
    outline: none;
    border-color: var(--accent-color);
    box-shadow: 0 0 0 3px rgba(52, 152, 219, 0.1);
}
.input-group input { flex: 1; }
button {
    cursor: pointer;
    border: none;
    border-radius: 6px;
    font-weight: 600;
    transition: all 0.2s ease;
}
.btn-action {
    background-color: #e2e6ea;
    color: #444;
    padding: 0 15px;
    white-space: nowrap;
    font-size: 0.85rem;
}
.btn-action:hover { background-color: #dbe2e8; color: #222; }
.btn-calc {
    width: 100%;
    background-color: var(--primary-color);
    color: white;
    padding: 12px;
    margin-top: 10px;
}
.btn-calc:hover { background-color: #34495e; }
.btn-submit {
    width: 100%;
    background-color: #27ae60;
    color: white;
    padding: 14px;
    font-size: 1.1rem;
    margin-top: 20px;
    box-shadow: 0 4px 6px rgba(39, 174, 96, 0.2);
}
.btn-submit:hover { background-color: #219150; transform: translateY(-1px); }
.section-title {
    margin-top: 25px;
    margin-bottom: 15px;
    font-size: 0.85rem;
    text-transform: uppercase;
    letter-spacing: 1px;
    color: #999;
```

```
            font-weight: 700;
        }
        input[readonly] { background-color: #f8f9fa; color: #777; }
        .doc-link-container {
            text-align: center;
            margin-top: -15px;
            margin-bottom: 25px;
        }
        .btn-doc {
            display: inline-block;
            text-decoration: none;
            color: var(--accent-color);
            background-color: transparent;
            border: 1px solid var(--accent-color);
            padding: 6px 15px;
            border-radius: 20px;
            font-size: 0.85rem;
            font-weight: 600;
            transition: all 0.2s ease;
        }
        .btn-doc:hover {
            background-color: var(--accent-color);
            color: #fff;
            box-shadow: 0 2px 5px rgba(52, 152, 219, 0.3);
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>3D Pay Test Formu</h2>
        <div class="doc-link-container">
            <a href="https://sanalpos.innova.com.tr/ziraatbankasi/Dosyalar/3DPay_Entegrasyon_Dokumani.pdf"
               target="_blank"
               class="btn-doc">
               3D PAY Entegrasyon Dokümanına buradan erişebilirsiniz
            </a>
        </div>
        <form id="testForm" method="POST"
action="https://sanalpos.innova.com.tr/ZIRAATBANK/VposWeb/v3/VposThreeDPay.aspx">

            <div class="section-title">Ödeme Bilgileri</div>
            <div class="form-group">
                <label>MerchantId</label>
                <input type="text" name="MerchantId" value="000000000281567" required />
            </div>
            <div class="form-group">
                <label>OrderId</label>
                <div class="input-group">
                    <input type="text" name="OrderId" id="orderIdField" placeholder="Sipariş Numarası" />
                    <button type="button" class="btn-action" onclick="generateOrderId()">Oluştur</button>
                </div>
            </div>
            <div class="form-group">
                <label>StoreType</label>
                <input type="text" name="StoreType" value="3d_pay" readonly />
            </div>
            <div class="form-group">
                <div style="display: flex; gap: 15px;">
                    <div style="flex: 1;">
                        <label>Para Birimi (949=TL)</label>
                        <input type="text" name="CurrencyCode" value="949" required />
                    </div>
                    <div style="flex: 1;">
                        <label>Tutar (1.00)</label>
                        <input type="text" name="CurrencyAmount" value="1" required />
                    </div>
                </div>
            </div>

            <div class="section-title">Güvenlik & Hash</div>
            <div class="form-group">
                <label>Rnd (Random Değer)</label>
                <div class="input-group">
                    <input type="text" name="Rnd" id="rndField" required readonly />
                    <button type="button" class="btn-action" onclick="generateRnd()">Yenile</button>
                </div>
            </div>
            <div class="form-group">
                <label>HashData</label>
                <input type="text" name="HashData" id="hashDataField" placeholder="Hesapla butonuna basın"
required readonly />
                <button type="button" class="btn-calc" onclick="calculateHash()">Hash Hesapla &
İmzala</button>
            </div>

            <div class="section-title">Dönüş URL'leri</div>
            <div class="form-group">
                <label>SuccessUrl</label>
                <input type="text" name="SuccessUrl" value="https://www.google.com/success" required />
```

```
            </div>
            <div class="form-group">
                <label>FailUrl</label>
                <input type="text" name="FailUrl" value="https://www.google.com/fail" required />
            </div>

            <input type="hidden" name="TransactionType" value="Sale" />
            <input type="hidden" name="TransactionDeviceSource" value="0" />
            <input type="hidden" name="ClientIp" value="127.0.0.1" />

            <div class="section-title">Kart Bilgileri (Opsiyonel)</div>
            <div class="form-group">
                <label>Pan (Kart No)</label>
                <input type="text" name="Pan" value="5549601963997012" />
            </div>
            <div class="form-group">
                <div style="display: flex; gap: 15px;">
                    <div style="flex: 1;">
                        <label>Expiry (YYYYAA)</label>
                        <input type="text" name="Expiry" value="202909" />
                    </div>
                    <div style="flex: 1;">
                        <label>Cvv</label>
                        <input type="text" name="Cvv" value="259" />
                    </div>
                </div>
            </div>

            <div class="section-title">CallbackType (Default GET)</div>
            <div class="form-group">
                <label>CallbackType</label>
                <input type="text" name="CallbackType" value="GET" />
            </div>

            <button type="submit" class="btn-submit">Ödemeyi Başlat (Gönder)</button>
        </form>
    </div>

    <script>
        window.onload = function() {
            generateRnd();
            generateOrderId();
        };

        function generateOrderId() {
            const date = new Date();
            const timeStr = date.toISOString().slice(0,19).replace(/[^0-9]/g, "");
            const randomSuffix = Math.floor(1000 + Math.random() * 9000);
            document.getElementById('orderIdField').value = "ORD-" + timeStr + randomSuffix;
        }

        function generateRnd() {
            const chars = '0123456789abcdef';
            let rnd = '';
            for (let i = 0; i < 32; i++) {
                rnd += chars[Math.floor(Math.random() * chars.length)];
            }
            document.getElementById('rndField').value = rnd;
        }

        function getAsciiBytes(str) {
            const bytes = new Uint8Array(str.length);
            for (let i = 0; i < str.length; i++) {
                const charCode = str.charCodeAt(i);
                bytes[i] = charCode > 127 ? 63 : charCode;
            }
            return bytes;
        }

        async function calculateHash() {
            const merchantId = document.querySelector('[name="MerchantId"]').value;
            const orderId = document.querySelector('[name="OrderId"]').value || "";
            const currencyCode = document.querySelector('[name="CurrencyCode"]').value;
            const currencyAmount = document.querySelector('[name="CurrencyAmount"]').value;
            const successUrl = document.querySelector('[name="SuccessUrl"]').value;
            const failUrl = document.querySelector('[name="FailUrl"]').value;
            const rnd = document.querySelector('[name="Rnd"]').value;

            const merchantPassword = prompt('Lütfen Merchant Password girin:');

            if (!merchantPassword) {
                alert('Şifre girilmedi, işlem iptal edildi.');
                return;
            }

            const amountLong = (parseFloat(currencyAmount) * 100).toFixed(0);

            const hashString = merchantId + orderId + currencyCode + amountLong + successUrl + failUrl +
rnd + merchantPassword;
```

```
            console.log("Hash String:", hashString);

            const data = getAsciiBytes(hashString);
            const hashBuffer = await crypto.subtle.digest('SHA-512', data);
            const hashArray = Array.from(new Uint8Array(hashBuffer));
            const hashBase64 = btoa(String.fromCharCode(...hashArray));

            document.getElementById('hashDataField').value = hashBase64;
        }
    </script>
</body>
</html>
```

11. HATA KODLARI

Kod	Açıklama	Çözüm
0000	İşlem başarılı	İşlem başarıyla tamamlandı
1001	Sistem hatası	Tekrar deneyin veya destek ile iletişime geçin
1006	Bu OrderId ile başarılı işlem mevcut	Yeni bir OrderId oluşturun
1047	Geçersiz tutar formatı	Tutarı "10.50" formatında gönderin
1050	CVV hatalı veya eksik	CVV kodunu kontrol edin (3-4 hane)
1051	Kart numarası hatalı	Kart numarasını kontrol edin (16-19 hane)
1052	Son kullanma tarihi hatalı	Tarihi YYYYMM formatında gönderin (örn: 202512)
1054	TransactionId hatalı	Geçerli bir TransactionId gönderin
1060	Hatalı taksit sayısı	Taksit sayısını kontrol edin (2-12 arası)
1097	Geçersiz CAVV değeri	3D Secure CAVV değerini kontrol edin
1098	Geçersiz ECI değeri	3D Secure ECI değerini kontrol edin
1100	Geçersiz kart markası	BrandName parametresini kontrol edin (100-500)
2200	İşlem yetkisi yok	İşyeri bu işlem için yetkilendirilmemiş
2202	İptal edilemez (Batch kapalı)	Gün sonu sonrası iptal yapılamaz, iade yapın
5001	İşyeri şifresi yanlış	MerchantPassword veya hash hesaplamasını kontrol edin
5002	İşyeri aktif değil	İşyeri hesabınızın aktif olduğunu kontrol edin
1061	Sipariş numarası kullanımda	Farklı bir OrderId kullanın
9001	Hash doğrulama hatası	Hash hesaplama algoritmasını ve parametreleri kontrol edin
9999	Bilinmeyen hata	Destek ekibi ile iletişime geçin