PayFields

API Specification

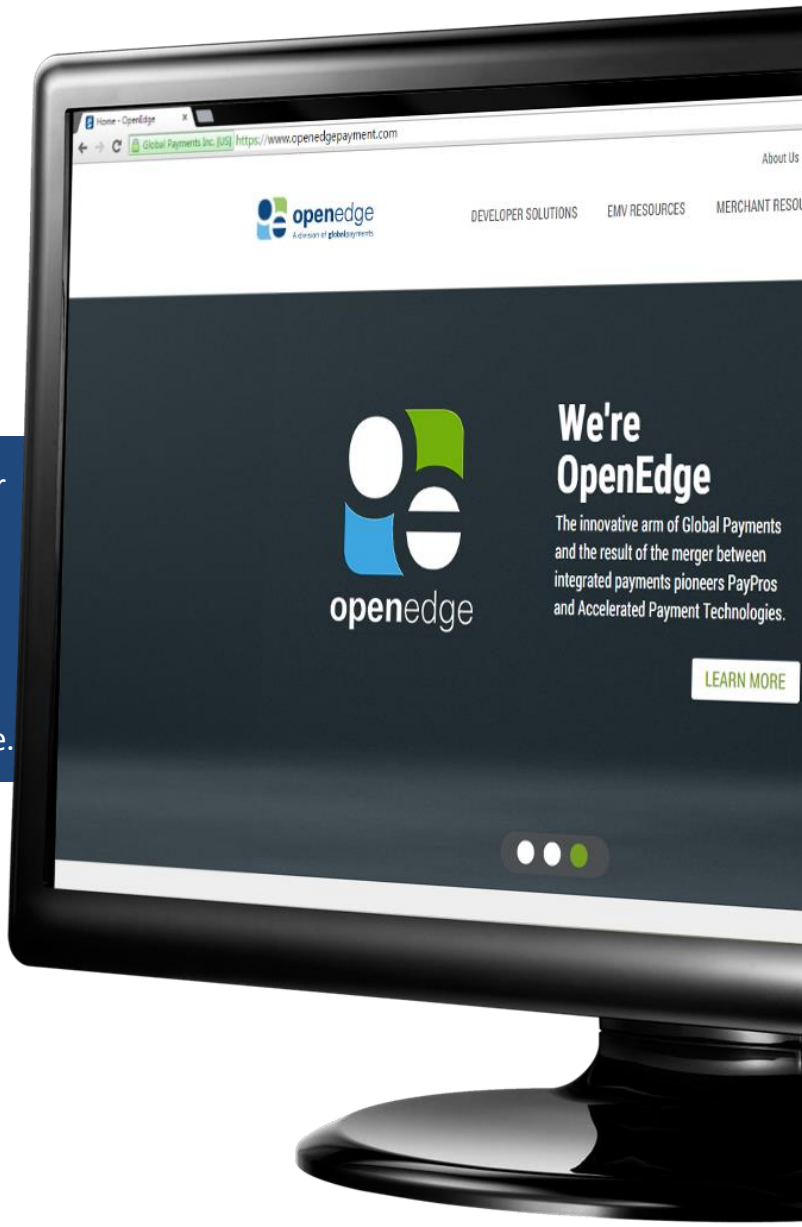


TABLE OF CONTENTS

OVERVIEW

PayFields provides the ability for payment card data to be collected in a secure manner by allowing the integrator to host discrete, customizable payment fields from OpenEdge's web server. Rendered fields can have their data modified at any time until transaction submission, allowing for modification of charge total and card data without the requirement for reloading the page.

The transaction processing API supports the following payment types:

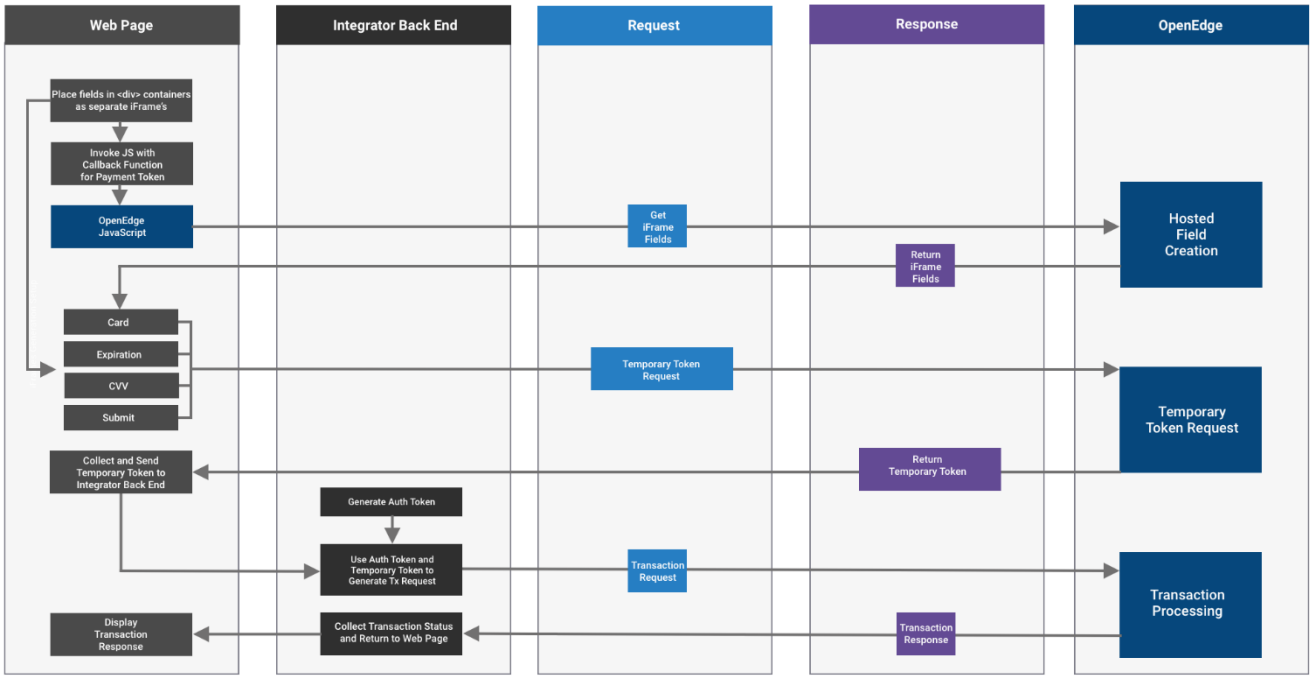
Payment Type Support	
Payment Types Supported	**Transaction Type**
Credit, Prepaid	Sale, Auth: 'Zero Dollar Auth', FSA / IIAS

The PayFields API is a HTTPS-based REST API which uses JSON payloads for requests and responses. JSON is, by definition, an unordered list of name-value pairs. Partners should parse the JSON by standard JSON object/libraries, which are order-agnostic. Partners should not build any one-off manual parsers for JSON, as the order of elements in the response can change over time, as new elements are added to the API response. Pending new elements in the API response, using one-off manual parsers for JSON will result in integration and transaction failures for partners and merchants.

The API endpoints are secured using one-way TLS (version 1.2).

The API supports cross-origin resource sharing (CORS), allowing integrating clients to securely interact with the API from a client-side web application. However, the API should not be invoked from any public website's client-side code, as it will expose the integrating client's assigned secret API Key.

WORKFLOW



PayFields Process Flow

- Card Entry Fields will be placed in containers as separate iFrames.
- Web page will invoke OpenEdge JavaScript with a Callback Function for returning a Payment Token.
- OpenEdge JavaScript will request to retrieve the iFrame fields and will return iFrame fields to be loaded.
- Card Data is entered on discrete payment fields rendered on web page.
- Upon clicking 'Submit' on web page, a 'Temporary Token' Request will be sent to OpenEdge and returned back to the web page.
- Web page will return the 'Temporary Token' and other details (like amount, invoice number etc.) captured from Web Page to the Integrator Back End.
- Integrator Back End will generate an AuthToken (secure processing credential) to process transaction once the 'Temporary Token' has been received.
- AuthToken and 'Temporary Token' with other transaction details are used to submit a transaction request.
- API Returns transaction response to Integrator Back End
- Integrator Back End handles response data to Integrators requirements and provides response back to web page.

JAVASCRIPT IMPLEMENTATION

This API uses hosted JavaScript in order to render discrete fields defined by HTML elements hosted on integrator web page by OpenEdge web servers. After the valid card data is entered in hosted fields and the 'Submit' button is clicked, the credit card is accepted and tokenized to be used with a Transaction Request.

Steps Required to Render Discrete Payment Fields

This API uses the following steps to render the discrete payment fields and retrieve the 'Temporary Token' for transaction processing.

Link the JavaScript to your web page.

Integrating clients will need to include the following JavaScript tag in your page <header>

```
<script
src= https://js.paygateway.com/secure_payment/v1/globalpayments.js>
</script>
```

Include HTML elements on your page for the discrete payment fields.

Include target credit card fields with unique ID's on your page to host fields from OpenEdge web servers.

```
<form id="payment-form" action="/charge" method="post">
  <!-- Other input fields to capture relevant data -->
  <label for="billing-zip">Billing Zip Code</label>
  <input id="billing-zip" name="billing-zip" type="tel" />

  <!-- Targets for the credit card form's fields -->
  <div id="card-number"></div>
  <div id="card-expiration"></div>
  <div id="card-cvv"></div>
  <div id="submit"></div>
</form>
```

Configure the Payments Object.

The Payments Object needs to be configured with your public API key and environment.

X-GP-Api-Key To consume the JavaScript, integrating clients need an API Key for both environments (test and production). The API Key is a secret and will be issued to clients respective to partner-product and major version consuming the API. Please contact OpenEdge Developer Services to obtain test and production API Keys.

```
GlobalPayments.configure({
  'X-GP-Api-Key': "<YOUR API KEY HERE>".
  'X-GP-Environment': "test" or "prod"
});
```

Call the UI form to render the input fields.

The UI form needs to be rendered with the input fields within the elements created.

```
var cardForm = GlobalPayments.ui.form({
  fields: {
    "card-number": {
      target: "#card-number"//renders field to <div id="card-number">
      placeholder: "•••• •••• •••• ••••", //Optional
    },
    "card-expiration": {
      target: "#card-expiration"
      placeholder: "MM / YYYY",
    },
    "card-cvv": {
      target: "#card-cvv"
      placeholder: "•••",
    },
    "submit": {
      text: "Submit",//Set text for Submit button
      target: "#submit"
    }
  },

  styles :{
    //your styles
  }
});
```

Create a callback to get the 'Temporary Token' and Submit to Integrators backend

After the user clicks the 'Submit' discrete field, the credit card will be accepted and a 'Temporary Token' will be received as a response to be used with a Transaction Request.

```
//For success

cardForm.on("token-success", (resp) => {
  // add payment token to form as a hidden input
  const token = document.createElement("input");
  token.type = "hidden";
  token.name = "temporary_token";
```

```
  token.value = resp.temporary_token;

  // submit data to the integration's backend for processing
  const form = document.getElementById("payment-form");
  form.appendChild(token);
  form.submit();
});
```

Handling failures from Tokenization Service

The Tokenization Service will return an error if there is any data missing from the form on submission.

```
cardForm.on("token-error", (resp) => {
  // show error to the consumer
}
```

Examples

Invalid API Key

```
{
 "error": {
  "code": "authentication_failed",
  "message": "Authentication Failed. Please retry with valid credentials",
  "info": "https://developers.api.dev.paygateway.com/errors"
 }
}
```

Handling Global Configuration Errors from JavaScript

The GlobalPayments global variable exposes an on function to attach event listeners for an internal event emitter. Global and/or parent window runtime errors are exposed through this, and error event handlers should follow the form:

```
type ErrorEventHandler = (error: IErrorEvent) => void;
```

where `IErrorEvent` follows the form:

```
{
  error: true,
  reasons: [{
    code,
    message,
  }],
}
```

Examples

```
GlobalPayments.on("error", (error) => {
  console.error(error);
});
```

No Gateway Available (Incorrect parameters passed to JavaScript)

```
{
   "error": true,
   "reasons": [
     {
       "code": "INVALID_CONFIGURATION",
       "message": "no gateway available"
     }
   ]
}
```

iFrame Loading Error

```
{
   "error": true,
   "reasons": [
     {
       "code": "ERROR",
       "message": "IframeField: target cannot be found with given selector"
     }
   ]
}
```

TRANSACTION PROCESSING WITH TEMPORARY TOKEN

After a temporary token has been generated and returned from the response object, the integrator will need to utilize the OpenEdge Transactions API to process any Sale or Auth* transactions using the temporary token as well as any supported follow-on transactions.

*Authorizations are limited to Zero Dollar Auth Only

OpenEdge Transactions API

JAVASCRIPT ERROR CODES

See the table below for a list of possible error scenarios and error messages returned by the JavaScript when attempting to generate a Temporary Token

error_scenario	error_code	error_message	error_description	data_path
Invalid card_number	invalid_input	Invalid input data.	Invalid data	/card/card_number
LUHN check	invalid_card	Invalid card_number. Check the card details or use a different card.		
Invalid expiry_month	invalid_input	Invalid input data.	Invalid data	/card/expiry_month
Invalid expiry_year	invalid_input	Invalid input data.	Invalid data	/card/expiry_year
Invalid card_security_card	invalid_input	Invalid input data.	Invalid data	/card/card_security_code
Internal Error	internal_error	Internal Server Error.		

FORM CUSTOMIZATION AND STYLING SAMPLES

PayFields are completely customizable to provide the look, feel and workflow desired to provide a seamless checkout experience. Samples provided below will give you an indication of some of the customization possible.

Match PayFields Style with Page

```
{
 "button, input": {
  "outline": "none",
  "border-width": "0",
  "letter-spacing": "2px"
 },
 "input, select": {
  "padding": "5px 10px",
  "font-size": "13px",
  "background": "#f3faff",
  "border": "1px solid #e0f1ff",
  "box-sizing": "border-box"
 },
 "input:focus": {
  "-webkit-box-shadow": "inset 0px 0px 3px 0px #b0dcff !important",
  "box-shadow": "inset 0px 0px 3px 0px #b0dcff !important",
  "border-color": "#b0dcff !important"
 },
 "button": {
  "background": "#125d85",
  "padding": "7px 40px",
  "color": "white",
```

```
  "font-size": "15px"
 },
 "button:active": {
  "filter": "brightness(112%)"
 }
}
```

Highlight Invalid Fields

```
{
 "input.invalid": {
  "-webkit-box-shadow": "inset 0px 0px 5px 0px #ff0a0a",
  "box-shadow": "inset 0px 0px 5px 0px #ff0a0a",
  "border-color": "#ff8282"
 },
 "input:focus": {
  "-webkit-box-shadow": "inset 0px 0px 3px 0px #b0dcff !important",
  "box-shadow": "inset 0px 0px 3px 0px #b0dcff !important",
  "border-color": "#b0dcff !important"
 },
 "input": {
  "border-width": "1px"
 }
}
```

Lock Submit Button

```
//These stylings will DISABLE the button
var disabledCSS = {
  "button" : {
      "color": "#9c9c9c",
     "pointer-events": "none",
     "cursor": "not-allowed"

     }
}

//These stylings will ENABLE the button
var enabledCSS = {
 "button" : {
      "color": "white",
      "pointer-events": "auto",
     "cursor": "pointer "
     }
}
```

Display Card Logo

```
{
 ".card-number.card-type-visa": {
  "background": "#f3faff url(" + imageURL + "logo-visa@2x.png) no-repeat
right",
  "background-position-y": "-6px",
  "background-size": "70px 75px"
 },
 ".card-number.card-type-mastercard": {
  "background": "#f3faff url(" + imageURL + "logo-mastercard@2x.png) no-
repeat right",
  "background-position-y": "-3px",
  "background-size": "60px 64px"
 },
 ".card-number.card-type-discover": {
  "background": "#f3faff url(" + imageURL + "logo-discover@2x.png) no-repeat
right",
  "background-position-y": "-11px",
  "background-size": "90px 96px"
 },
 ".card-number.card-type-jcb": {
  "background": "#f3faff url(" + imageURL + "logo-jcb@2x.png) no-repeat
right",
  "background-position-y": "-6px",
  "background-size": "48px 82px"
 },
 ".card-number.card-type-amex": {
  "background": "#f3faff url(" + imageURL + "logo-amex@2x.png) no-repeat
right",
  "background-position-y": "-12px",
  "background-size": "55px 98px"
 }
}
```

Display Field Errors on Submit

```
cardForm.on('token-error', function (response) {
let error = response.error;
let validationDataset =  [
     {
          isValid: (error.code === 'invalid_card') ? false : true,
          searchTerm: 'card_number',
          elementID: 'card-number',
          message: 'Invalid card number'
     },
     {
          isValid: true,
          searchTerm: 'expiry',
          elementID: 'card-expiration',
```

```
            message: 'Invalid expiration date'
      },
      {
            isValid: true,
            searchTerm: 'card_security_code', //CVV
            elementID: 'card-cvv',
            message: 'Invalid CVV'
      },
];

for (let validationData of validationDataset) {
      if (error.detail) {
            for (let detail of error.detail) {
                  const data_path = detail.data_path || '';
                  const description = detail.description || '';
                  if (data_path.search(validationData.searchTerm) > -1
                        || description.search(validationData.searchTerm) > -1)
{
                        validationData.isValid = false;
                  }
            }
      }
      if (!validationData.isValid) {
            showError(validationData.elementID, validationData.message);
      }
}
});

function showError(elementID, msg) {
      let numberErrorMsg = document.querySelector('#' + elementID + '
.field-message');
      numberErrorMsg.style.display = 'block';
      numberErrorMsg.textContent = msg;
}
```

CHANGE HISTORY

Change History		
Date	**Author**	**Reason for Update**
07/10/2019	NSC	• Initial publication.
08/15/2019	MB RK	• Added JavaScript Error Codes • Added Form Customization and Styling samples