# Automatic cardtype detection – with some debug

This example (not nice, but useful) will show how to configure PAYONE hosted-iFrame mode with cardtype selection in a PAYONE iFrame and automatic cardtype detection enabled. The field for cardtype selection can be used to override the automatic detection.

Simply change:

- set array “supportedCardtypes” to your cardtypes you’d like to process
- config.fields.*.style to your CSS layout
- request.*-fields to your mid, aid, portalid, mode, encoding and hash value

and finally

- initiate a preauthorization / authorization via received “pseudocardpan”.

```
<!DOCTYPE html>
<html>
    <head lang="en">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link href="www.your-url.com/static/client_api.css" rel="stylesheet" type="text/css" media="screen" />
    <link rel="shortcut icon" type="image/x-icon" href="www.your-url.com/static/favicon.ico" />
    <meta charset="UTF-8">
    <title>Client API with the new hosted iFrames</title>
</head>
<body>
<h1>PAYONE Client API with new hosted iFrames</h1>
<h2>The hosted iFrames beside two regular input fields</h2>
<form name="paymentform" action="#" method="post">
    <fieldset>
        <div id="loading" style="padding: 50px; border: 5px solid orange;">
            <p style="font-size: 2em; color: orange; text-align: center;">iFrames still loading...</p>
        </div>

        <input type="hidden" name="pseudocardpan" id="pseudocardpan">
        <input type="hidden" name="truncatedcardpan" id="truncatedcardpan">
        <input type="hidden" name="cardtypeResponse" id="cardtypeResponse">
        <input type="hidden" name="cardexpiredateResponse" id="cardexpiredateResponse">

        <label for="cardtype">Card type:</label>
        <span class="inputIframe" id="cardtype"></span>

        <label for="cardpanInput">Cardpan:</label>
        <span class="inputIframe" id="cardpan"></span> <!-- Container for PAYONE-Script -> PAYONE will place an
iFrame in here -->

        <label for="cvcInput">CVC:</label>
        <span class="inputIframe" id="cardcvc2"></span>

        <label for="expireInput">Expire date (mm/yyyy):</label>
        <span class="inputIframe" id="expireInput">
            <span id="cardexpiremonth"></span>
            <span id="cardexpireyear"></span>
        </span>

        <label for="firstname">Firstname:</label>
        <input id="firstname" type="text" name="firstname" value="" placeholder="Firstname">

        <label for="lastname">Lastname:</label>
        <input id="lastname" type="text" name="lastname" value="" placeholder="Lastname">

        <div id="error"></div>
        <br />
        <input id="submit" type="button" value="Run credit card check" />
    </fieldset>
</form>
<h2>The JSON received from the server</h2>
<div id="jsonResponse"><pre id="jsonResponsePre">Nothing received yet.</pre></div>
<h2>Autodetection callback result</h2>
<div id="autodetectionResponse"><pre id="autodetectionResponsePre">Nothing received yet.</pre></div>
```

```
<h2>Set card type manually</h2>
<div class="maxWidth">
    <p class="infoText">
        It is possible to set the card types manually. It is not required to use the hosted iFrame with the
card type
        drop down select. Therefore you can use the following two buttons to change the card type either to V
or to J.
        This can be used to test detailed configuration options with the cardcvc2-iFrame.
    </p>
</div>
<div class="maxWidth">
    <p class="infoText">
        <input type="text" id="manCardType" value="" maxlength="1" style="width: 50px; display: inline;" />
        <button id="setCardTypeManually">Set card type manually to the value on the left</button>
    </p>
</div>
<div id="setToVDiv"><button id="setToV">Set Credit Card to V manually</button></div>
<div id="setToJDiv"><button id="setToJ">Set Credit Card to J manually</button></div>
<div class="maxWidth">
    <p class="infoText">
        This button uses the setCardType() with an empty string. This should not trigger a runtime error on the
JavaScript-
        console of your browser.
    </p>
</div>
<div class="maxWidth">
    <p class="infoText">
        This button sends valid JSON to change the credit card to 'M' as the setCardType() method would do.
However
        we do not send the origin parameter within the JSON so it should be ignored. Attention: The card type
selector
        would not be changed, even if the request is valid.
    </p>
</div>
<div class="maxWidth">
    <p class="infoText">
        In some exceptional cases the iFrames that we create during runtime can receive messages by
postMessage() which
        are not meant for them. This button sends some crap string that does not represent a JSON string. This
should not
        deal any damage to the JavaScript runtime within payone_hosted*.js or payone_iframe*.js.
    </p>
</div>
<script>

    var request,
        supportedCardtypes = ["#", "V", "M", "J", "U", "P"],
        config;

    /*

    // Some overwriting and placeholder enabling examples.

    // Example for enabling placeholder attribute in <input type="text" /> fields.
    // Defaults to empty, so placeholder will not be applied.
    Payone.ClientApi.Language.de.placeholders.cardpan = 'Kartennummer';
    Payone.ClientApi.Language.de.placeholders.cvc = 'CVC';
    // Please keep in mind: placeholders are not supported for input type “select”.
    Payone.ClientApi.Language.de.placeholders.expireMonth = '(MM)';
    Payone.ClientApi.Language.de.placeholders.expireYear = '(JJJJ)';

    // Example for overwriting error messages in english.
    Payone.ClientApi.Language.en.invalidCardpan: "your error text";
    Payone.ClientApi.Language.en.invalidCvc: "your error text";
    Payone.ClientApi.Language.en.invalidPanForCardtype: "your error text";
    Payone.ClientApi.Language.en.invalidCardtype: "your error text";
    Payone.ClientApi.Language.en.invalidExpireDate: "your error text";
    Payone.ClientApi.Language.en.invalidIssueNumber: "your error text";
    Payone.ClientApi.Language.en.transactionRejected: "your error text";

    // Example for overwriting default month i18n in english (instead of 1, 2, 3 etc.)
```

```
    Payone.ClientApi.Language.en.months.month1 = "January";
    Payone.ClientApi.Language.en.months.month2 = "February";
    Payone.ClientApi.Language.en.months.month3 = "March";

    */

    config = {
        fields: {
            cardpan: {
                selector: "cardpan",
                style: "font-size: 14px; border: 1px solid #000;",
                styleFocus: "font-size: 12px; border: 1px solid orange;",
                type: "input"
            },
            cardcvc2: {
                selector: "cardcvc2",
                type: "password",  // Could be "text" as well.
                style: "font-size: 14px; border: 1px solid #000;",
                size: "4",
                // This is a configuration example for something like:
                // * Amex: Exactly 4 digits.
                // * Visa: Exactly 3 digits.
                // * JCB: No CVC, input disabled.
                // * Mastercard: Not configured, up to maxlength digits or none.
                // Notice that our validation code should be stable with case insensitive configurations made by
                // the merchant. This is why we have a lower case "a" and an upper case "V" and "J" for this
demo
                // configuration.
                maxlength: "4",
                length: { "a": 4, "V": 3, "J": false }
            },
            cardexpiremonth: {
                selector: "cardexpiremonth",
                type: "text", // select (default), text, tel (for alternative keyboards on mobile devices)
                size: "2",
                maxlength: "2",
                iframe: {
                    width: "40px"
                },
                style: "font-size: 14px; width: 30px; border: solid 1px #000; height: 22px;"
            },
            cardexpireyear: {
                selector: "cardexpireyear",
                type: "text", // select (default), text, tel (for alternative keyboards on mobile devices)
                iframe: {
                    width: "60px"
                },
                style: "font-size: 14px; width: 50px; border: solid 1px #000; height: 22px;"
            },
            cardtype: {
                selector: "cardtype",
                cardtypes: supportedCardtypes
            }
        },
        defaultStyle: {
            input: "font-size: 1em; border: 1px solid #000; width: 175px;",
            select: "font-size: 1em; border: 1px solid #000;",
            iframe: {
                height: "22px",
                width: "180px"
            }
        },

        autoCardtypeDetection: {
            supportedCardtypes: supportedCardtypes,
            callback: function(detectedCardtype) {
                // Individual callback code starts here:
                document.getElementById('autodetectionResponsePre').innerHTML = detectedCardtype;
                // Individual callback code ends.
            } //,
            // deactivate: true // To turn off automatic card type detection.
```

```
            // Must be explicitly set to boolean true.
        },

        events: {
            // Javascript callback function will be triggered once “hosted iFrames” are completely loaded
            rendered: function () {
                document.getElementById('loading').remove()
            }
        },

        language: Payone.ClientApi.Language.de, //, // Language to display error-messages (default: Payone.
ClientApi.Language.en)
        error: "error" // area to display error-messages (optional)
    };

    request = {
        request: 'creditcardcheck',                 // fixed value
        responsetype: 'JSON',                       // fixed value
        mode: 'live',                               // desired mode
        mid: '10001',                               // your MID
        aid: '10002',                               // your AID
        portalid: '2000002',                        // your PortalId
        encoding: 'UTF-8',                          // desired encoding
        storecarddata: 'yes',                       // fixed value
        // hash calculated over your request-parameter-values (alphabetical request-order) plus PMI portal key
        // hash:
'1cf456bf692453613ebb992a3fb859cc347ddc7e94e2ca764efbe8b0089de6964ab1266df0831e59de89dc5291070fe7'
        hash: '5c4014cebeb361d9e186fd42c810b9b1'     // see Chapter 3.1.5.3
    };
    var iframes = new Payone.ClientApi.HostedIFrames(config, request);

    document.getElementById('setCardTypeManually').onclick = function() {
        iframes.setCardType(document.getElementById('manCardType').value);
    };

    document.getElementById('setToV').onclick = function() {
        iframes.setCardType('V');
    };

    document.getElementById('setToJ').onclick = function() {
        iframes.setCardType('J');
    };

    // Function called by submitting PAY-button
    function pay() {

        if (iframes.isComplete()) {
            // Perform "CreditCardCheck" to create and get a PseudoCardPan; then call your function
"payCallback"
            iframes.creditCardCheck('payCallback');
        } else {
            alert("Not complete. Nothing done.");
        }
    }

    document.getElementById("submit").onclick = function () {
        pay();
    };

    function payCallback(response) {

        console.debug(response);
        if (response.status === "VALID") {
            document.getElementById("pseudocardpan").value = response.pseudocardpan;
            document.getElementById("truncatedcardpan").value = response.truncatedcardpan;
            document.getElementById("cardtypeResponse").value = response.cardtype;
            document.getElementById("cardexpiredateResponse").value = response.cardexpiredate;
        }

        if (typeof response === 'object') {
```

```
            var responseAsString = 'time: ' + new Date().getTime() + "\n";
            for (var key in response) {

                if (response.hasOwnProperty(key)) {
                    responseAsString += key + ': ' + response[key] + "\n";
                }
            }

            document.getElementById('jsonResponsePre').innerHTML = responseAsString;
        }
    }

</script>
</body>
</html>
```