Recurring Payment - Variable Amount & Variable Duration

The solution allows merchants to retain customer payment information (with consent) for future transactions, reducing checkout friction and improving conversion rates.

There are 2 use cases that require storing of credentials:

- **Unscheduled Customer Initiated Transactions (CITs):** CITs occur when customers actively authorize a payment during checkout. To streamline the process for returning customers, you can securely store payment credentials (with consent), enabling faster "one-click" transactions for future purchases.
- **Unscheduled Merchant Initiated Transactions (MITs):** MITs occur when you initiate a payment without direct participation of the customer. You can initiate an unscheduled MIT based on an explicit prior authorization from the customer to store and use their credentials.

Implementation workflow

You can setup Unscheduled CITs and MITs with by following 3 simple steps:

1. Initial transaction with CIT/MIT intent
2. Store credentials securely
3. Subsequent transactions

Unscheduled CITs

1. Initial transaction

Setup intent for an unscheduled CIT by passing the below information in the `credentialOnFile` object in the payment request:

```
{
  ...
  "credentialOnFile": {
    "type": "UNSCHEDULED",
    "initialPayment": true,
    "unscheduled": {
        "subType": "CIT"
    }
  }
}
```

2. Credential storage

Securely store the credentials based on the integration type upon receiving successful payment response to the initial transaction:

Integration type	Action

Hosted forms – Hosted Payment Page or card Hosted forms	Fetch the `pseudoCardNumber` by calling Retrieve payment details by payment ID and store it. `{ ... "paymentMethods": { "type": "CARD", "card": { "cardholderName": "John Doe", "pseudoCardNumber": "01234567890124444", "first6Digits": 555555, "last4Digits": 4444, "expiryDate": "01.01.2028", "schemeReferenceId": "534525242342", ..... ..... } } }`
Direct Integration	Store either: • Clear card data securely that you already have during the payment process. • `pseudoCardNumber` received in payment response if you have the service enabled.

3. Subsequent transactions

Submit the subsequent payment request with request parameters depending on the type of your integration

Integration type	Payment parameters

<table>
<tr>
<td>Hosted forms – Hosted Payment Page or card Hosted forms</td>
<td>
Submit the <code>credentialOnFile</code> object as below:
<pre>
{
  ...
  "credentialOnFile": {
      "type": "UNSCHEDULED",
      "initialPayment": false,
      "unscheduled": {
          "subType": "CIT"
      }
  }
}
</pre>
Pass the Pseudo card number in <code>prefillInfo</code> object and <code>schemeReferenceId</code> returned previously in the response of initial transaction, for a faster checkout.
<pre>
{
  ...
  "paymentMethods": {
    "type": "CARD",
    "card": {
      "prefillInfo": {
          "number": "01234567890124444",
          "cardholderName": "John Doe",
          "expiryDate": "01.01.2028",
          .....
          .....
      },
      "schemeReferenceId": "534525242342"
    }
  }
}
</pre>
</td>
</tr>
<tr>
<td>Direct Integration</td>
<td>

Submit the <code>credentialOnFile</code> object as below:
<pre>
{
  ...
  "credentialOnFile": {
      "type": "UNSCHEDULED",
      "initialPayment": false,
      "unscheduled": {
          "subType": "CIT"
      }
  }
}
</pre>

Pass the saved credentials or pseudo card number in <code>paymentMethods.card</code> object and <code>schemeReferenceId</code> returned previously in the response of initial transaction:
<pre>
"paymentMethods": {
  "type": "CARD",
  "card": {
    "number": "5555555555554444",
    "cardholderName": "John Doe",
    "expiryDate": "01.01.2028",
    "securityCode": "123"
    .....
    .....
  }
}
</pre>

</td>
</tr>
</table>

Recommendation:

Use Customer Vault instead of prefillInfo, if you are integrated using hosted forms for a better customer experience and reduced integration efforts.

Unscheduled MITs

1. Initial transaction

Setup intent for an unscheduled MIT by passing the below information in the `credentialOnFile` object:

```
{
  ...
  "credentialOnFile": {
    "type": "UNSCHEDULED",
    "initialPayment": true,
    "unscheduled": {
    "subType": "MIT"
    }
  }
}
```

2. Credential storage

Securely store the credentials and `schemeReferenceId` based on the integration type upon receiving successful payment response to the initial transaction:

Integration type	Action
Hosted forms – Hosted Payment Page or card Hosted forms	Fetch the `pseudoCardNumber` by calling Retrieve payment details by payment ID and store it. `{ ... "paymentMethods": { "type": "CARD", "card": { "cardholderName": "John Doe", "pseudoCardNumber": "01234567890124444", "first6Digits": 555555, "last4Digits": 4444, "expiryDate": "01.01.2028", "schemeReferenceId": "534525242342", ..... ..... } } }`
Direct integration	Store either: • Clear card data securely that you already have during the payment process. • `pseudoCardNumber` received in payment response if you have the service enabled.

3. Subsequent transactions

Subsequent transactions for an unscheduled MIT should always be sent via the direct integration.

- Pass the below data in `credentialOnFile` object for susbequent MITs:

```
{
  ...
  "credentialOnFile": {
    "type": "UNSCHEDULED",
    "initialPayment": false,
    "unscheduled": {
        "subType": "MIT"
    }
  }
}
```

- In addition to the above pass the saved credentials and `schemeReferenceId` returned previously in the response of initial transaction:

```
{
  ...
  "paymentMethods": {
    "type": "CARD",
    "card": {
      "number": "5555555555554444", // clear card number or pseudo card number
      "cardHolderName": "John Doe",
      "expiryDate": "202506",
      "securityCode": "123",
      "schemeReferenceId": "534525242342"
      .....
      .....
    }
  }
}
```