

PayComplete

# Connect Cube

A Whitepaper by PayComplete

Developing Financial Transaction Kiosks:  
Delivering Better Solutions to Market, Faster





## Deliver innovative devices to market faster

Businesses are increasingly recognising how kiosks can enhance day-to-day operations, enabling them to scale services up or down with far greater agility to match fluctuations in demand. This model has long been embedded within payment systems across the public transport sector, from parking payments to fare collection, where speed, reliability, and user convenience are critical.

Today, the public's familiarity and comfort with self-service technology continues to grow. As a result, organisations are leveraging kiosk solutions to deliver more cost-effective, efficient, and time-saving services, while optimising staff resources.

This evolving relationship between people and technology presents service providers with new opportunities to enhance customer experiences, modernise infrastructure, and expand their kiosk capabilities with added features and services, all while improving operational performance.

## The Challenge

Today's consumers have high expectations of self-service technology, particularly in relation to speed, ease of use, reliability, and overall functionality.

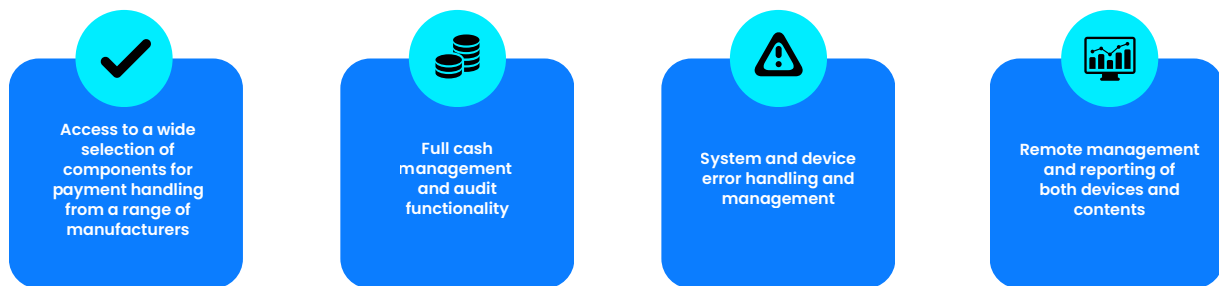
Behind the user experience, however, sits a complex technological ecosystem. From component selection and system integration, to secure payment handling, cash balancing, engineering support, remote management, and performance reporting, every element must work seamlessly together.

Alongside these operational demands, organisations must also keep pace with continuous advancements in market technology. Collectively, these factors make the product development lifecycle both significant in scope and costly to deliver, often extending the time required to bring new solutions to market.

## Connect Cube

The ConnectCube platform offers a radically different approach to kiosk development. So radically different, in fact, that this paper will show that with only limited code you can create a kiosk platform to manage financial transactions.

**Some of the key functions supported include:**



ConnectCube enables transactional device manufacturers to rapidly develop new solutions through the provision of an API and infrastructure management solution that sits with the operating system.

With an extensive library of components supported, this transaction-centric operating system governs the core transaction concepts, such as purchases, payments, deposit, dispense, refill, and more, that drive kiosk operations.

To demonstrate the simplicity of process and potential speed with which a solution can be developed, the code used for a basic payment kiosk solution supporting a number of features is shown in the following pages.



The process shown on the following pages uses the SDK rather than API, but reflects the flexibility and simplicity of the integration and development process within ConnectCube and the options available to kiosk developers.

**Our example shows a basic payment solution that uses some of the features of Cube and will:**



Present a pin pad where the user can input a payment amount



Process the payment - for whichever acceptance devices are connected



Dispense any required change, assuming dispensing devices are connected and have stocks



Print a receipt



Create a QR coded credit voucher if change not available

In addition to the functions included within our simple demonstration, here are a few other things that ConnectCube can manage, so that the kiosk application doesn't have to:



VAT calculations

---



Payment types (cash/cashless loyalty system/tokens/ credits)

---



Split payments between tender types (e.g. part cash and part cashless)

---



Overpayment handling

---



Reporting of accounting data (including products sold and usage)

---



Handling of price tables (nominal prices, discount prices, prices per tender type)

## Process Flow

This example is designed to simply and easily explain the potential processes and key tasks within the developers kiosk application.

**These colour coded sections reflect the functionality in the complete code example shown on page 7.**

### Start up

Implement the ConnectCube modules.

### Setting the Process Parameters

In our example, we have hardcoded the currency (to EUR) and set the VAT rate.

### Create a UI

Our sample uses a simple keypad entry for the payment. The kiosk developer would focus on UI/UX here.

### Process the Payment

Control and manage payment made by the user, with the ConnectCube processes managing the payment acceptance cycle.

### Supply the Product

Allow the client application to deliver any paid goods - such as producing tickets, sending data to other delivery systems.

### Housekeeping and Finish

Produce a transaction receipt. Creation of any credit voucher in the event of failed change delivery etc.

## Start up

```

package com.paycomplete.intertraffic;

import java.util.Arrays;
import java.util.Currency;
import java.util.HashMap;

import com.paycomplete.sdk.oj.IOJFlowSession;
import com.paycomplete.sdk.oj.IOJProduct;
import com.paycomplete.sdk.oj.IOJPurchaseManager;
import com.paycomplete.sdk.oj.IOJRegistrar;
import com.paycomplete.sdk.oj.IOJUIManager;
import com.paycomplete.sdk.oj.OJPrinterException;
import com.paycomplete.sdk.oj.OJPrinter.OJStandardReceiptType;
import com.paycomplete.sdk.plugin.IOJFlow;
import com.paycomplete.sdk.plugin.IOJPaymentListener;
import com.paycomplete.sdk.plugin.IOJPlugin;
import com.paycomplete.sdk.util.OJMoney;
import com.paycomplete.sdk.util.OJMoney.CurrencyFormat;
import com.paycomplete.sdk.util.OJMoney.OJMoneyType;
import com.paycomplete.sdk.util.OJMoneyBag;
import com.paycomplete.sdk.util.OJPriceTableEntry;

/**
 * This example shows the payment functionality as a Java plugin. The same functionality
 * is also available as a REST API.
 */
public class PaymentHandler implements IOJPlugin
{

```

## Setting the Process Parameters

```

private static final Currency CURRENCY = Currency.getInstance("EUR");
private static final float VAT = 20f; // 20% VAT

@Override
public void initialize( IOJRegistrar registrar )
{
    try
    {
        registrar.registerBootFlow( new PaymentFlow() );
    }
    catch ( Exception e )
    {
    }
}

private class PaymentFlow implements IOJFlow
{
    private OJMoneyBag amountPaid = new OJMoneyBag();
    private OJMoneyBag amountToPay = new OJMoneyBag();
    private boolean isCancelPossible = false;

    @Override

```

## Create a UI

```

public void run ( IOJFlowSession session )
{
    while ( true )
    {
        Integer amount = session.getUIManager().inputNumber( "Enter amount to pay in " + CURRENCY.getCurrencyCode(),
            0, false, 0, Integer.MAX_VALUE, CURRENCY.getDefaultFractionDigits() );

        if ( amount == null )
        {
            break; // Return control to the standard Connect on-Device flows
        }

        try
        {

```

## Process the Payment

```

session.getUserManager().loginDefaultUser();

amountToPay = new OJMoneyBag ( Arrays.asList( new OJMoney ( CURRENCY , amount, OJMoneyType.NOTE, 1 ) ) );
amountPaid = new OJMoneyBag();
isCancelPossible = false;
showPaymentScreen( session );

final IOJPurchaseManager purchase = session.getPurchaseManager();

purchase.beginPurchase ( CURRENCY , VAT );

IOJProduct product = purchase.createProduct( "Example group", "Example product",
    new OJPriceTableEntry( amount, amount), // Cash price
    new OJPriceTableEntry( amount, amount), // Credit card price
    new OJPriceTableEntry( amount, amount) ); // Loyalty card price

session.getUIManager().addKeyListener( ( key ) ->
{
    if ( key == IOJUIManager.OJKey.KEY_CANCEL )
    {
        purchase.cancelPayment();
    }
} );
boolean completed = purchase.doPayment( Arrays.asList( product ),
    Arrays.asList( IOJPurchaseManager.PaymentType.values() ),
    new IOJPaymentListener()
{
    public void setCancelPossible( boolean cancelPossible )
    {
        isCancelPossible = cancelPossible;
        showPaymentScreen( session );
    }
} );

@Override
public void updatePayment( OJMoneyBag paid )
{
    amountPaid = paid;
    showPaymentScreen( session );
}

@Override
public void externalPayment()
{
    session.getUIManager().showTextScreen( "Please make your payment", "Follow instructions on terminal", null, null );
}

@Override
public void externalPaymentAborted ( AbortReason reason )
{
    session.getUIManager().showTextScreen( "Please make your payment", "Aborted", null, null );
}
} );

session.getUIManager().removeAllKeyListeners();
isCancelPossible = false;
showPaymentScreen( session );

if ( completed )
{

```

## Supply the Product

```

        session.getUIManager().showTextScreen("", "Supplying products...", null, null );

        // TODO: Supply the products (this is managed by the application itself)

        product.setDelivered( true, null, "", -1);
    }

    session.getUIManager().showTextScreen("", "Supplying change...", null, null );
    purchase.payChange();

    purchase.endPurchase();

    session.getUIManager().showTextScreen("", "Printing receipt...", null, null );
    try
    {
        session.getDeviceManager().getPrinter().printStandardReceipt( OJStandardReceiptType.PURCHASE );
    }
    catch ( OJPrinterException e )
    {
    }

    // Print a voucher if there are remaining change that wasn't paid out
    if ( session.getUserManager().getCurrentUser().getCredits().getMoney().stream().mapToInt(
        m -> m.getNumberofItems() ).sum() > 0 )
    {
        purchase.giveVoucher( null, VAT ); // This returns a reference that we can choose to put in a database
        session.getUIManager().showTextScreen("", "Printing voucher...", null, null );
        try
        {
            session.getDeviceManager().getPrinter().printStandardReceipt( OJStandardReceiptType.CREDITS_REFUND );
        }
        catch ( OJPrinterException e )
        {
        }
    }
}

```

## Housekeeping and Finish

```

    }
    finally
    {
        session.getUserManager().logout();
    }
}

private void showPaymentScreen( IOJFlowSession session )
{
    HashMap<IOJUIManager.OJKey, String> keys = new HashMap<>();
    if ( isCancelPossible )
    {
        keys.put( IOJUIManager.OJKey.KEY_CANCEL, "Cancel" );
    }

    session.getUIManager().showTextScreen( "Please make your payment", "Paid: " +
        amountPaid.getFormattedValue( session.getSystemManager().getLocale(), "", CurrencyFormat.CURRENCY_CODE, false,
        false ) +
        "<br>To pay: " +
        amountToPay.getFormattedValue( session.getSystemManager().getLocale(), "", CurrencyFormat.CURRENCY_CODE,
        false, false ),
        null, keys );
}
}
}
}

```



## Benefits of the Cube Platform

The application process within the ConnectCube development solution is simple and effective allowing a fully functional payment application, albeit basic, to be created in 2 pages of code. Behind this simple solution is a powerful engine that enables your application to operate with a wide range of readily available payment devices, giving the maximum flexibility for kiosk configuration. Supported equipment includes a broad range of devices including cash handling, card readers, printers bar code and QR scanners, screens, through to complete payment systems.

**The ConnectCube platform manages the internal financial transactions including keeping track of current balances across all devices within the kiosk.**

In addition, the transactional data, device status and health (for all key peripherals), alerts, remote service and transactional reporting are automatically communicated to the cloud based central management system. With built in Zero Touch Service methodologies, the remote system allows operators to manage and maintain a wide estate of systems simply and efficiently, whilst allowing for components, and feature updates to be carried out remotely.

## Faster Design, Lower Costs

The ConnectCube platform creates flexibility in the use of device components, generating benefits for both the technical design team – access to the widest range and the latest solutions – and for the commercial team – easy ‘dual sourcing’ of key devices elements without the need to have additional engineering work carried out to bring onboard new suppliers. With this flexibility the ConnectCube platform allows the deployment of large kiosk networks, with a ‘ready to go’ cloud management system, allowing developers to both support their estate and to act as a tool they can offer to their end users for local estate management.

## Take the Next Step...

Contact PayComplete to gain access to the developer portal, see how quick and easy it can be to get a new kiosk with financial transactions, and start your next project now.

 [info@paycomplete.com](mailto:info@paycomplete.com)

## Excellence through Heritage and innovation

PayComplete is the global leader in cash management solutions, combining best in class hardware solutions with game changing software, unifying cash management with other key payments and operational systems. Dedicated to innovating self-service experiences and operations for both consumers and employees, The PayComplete IoT platform is made up of an adaptable set of SaaS and machine software, intelligent devices, and professional, technical and merchant services.

PayComplete Connect unifies the management of transactions, users, devices, and data across the enterprise, bringing digital precision to cash transactions and systems. PayComplete serves a broad range of industries, including retail, transportation, financial services, cash centers, mints and more. Industry leaders work with PayComplete to make their cash transaction-based businesses more innovative, agile, and efficient.



## Our Industries



### Retail

A wide range of merchants are using PayComplete to develop new self-service solutions that help them improve the customer experience, boost in-store efficiency, and make better decisions for their business.



### Cash Centers & Mints

With PayComplete's digital approach to cash management, cash processors are modernizing how coins move through cash centers and mints. The result is greater efficiency, increased recirculation, and lower minting costs.



### Mass Transit & Parking

Whether by car, public transport, or new mobility services, traveller payment convenience is improved throughout the journey with PayComplete solutions, along with the operational efficiencies of service providers.



### Hospitality & Leisure

PayComplete empowers hospitality and leisure hosts to provide great experiences for returning guests, and also operate efficiently amidst staff shortages and margin squeezes.



### OEM's

PayComplete empowers device OEMs to develop and deploy smart, connected, and differentiated transactional devices faster by leveraging a broad range of components, software, and services.



### Banking & Financial Services

As the number of staffed branches decreases, PayComplete provides self-service solutions that give banking customers more options. Cash management solutions improve branch operations and broaden treasury portfolios.

# We can help to Deliver innovative devices to market faster

## Meet the unique needs of any transaction experience with PayComplete's ConnectCube!

ConnectCube gives device manufacturers the competitive advantage of speed, flexibility, and built-in transaction expertise to help you meet the unique demands of your customers.

An adaptable combination of software, hardware, and extensibility, ConnectCube can be used across a broad category of industries.

### Contact us to find out more:

[info@paycomplete.com](mailto:info@paycomplete.com)

[www.paycomplete.com](http://www.paycomplete.com)

