IXC

Payment integration kickoff checklist · domestic PG and simple pay

Preparing amount validation, state transitions, retries, and operational handover

A practical IXC resource for adoption decisions and delivery

Published 2026.09 · English edition · PDF + Excel worksheets

Using this resource

Sets the contract, environment, and state rules that payment development needs, and records the validation results of success and failure scenarios.

Backend developers, PMs, and QA and operations staff starting a payment integration

Contents

01 Confirm kickoff conditions by environment and payment method 3

02 Let the server decide amounts and states 4

03 Handle retries and duplicate webhooks 5

04 Validate normal and exception paths together 6

05 Make sure operations staff can find transactions 7

How to complete and reuse it

Read the criteria in this PDF, then fill the blank fields with your actual conditions and anything still unconfirmed. The Excel file for the same practice area has a worksheet for each resource. Keep the worked examples separate from real customer cases and confirmed quotes.

With attribution, you may quote and adapt this material for internal review and your own work. The latest files are at www.ixc.co.kr/en/assets.

Confirm kickoff conditions by environment and payment method

Separate what a test integration needs from the conditions for starting real payments in production. Confirm which features work with the documentation test keys and which need merchant-specific test keys. Track production key issuance and payment method activation separately, as launch readiness items.

Payment integration environment sheet

Preparation item	Test environment	Production environment	Owner and status
Merchant ID and key storage location			
Contracted and active payment methods			
SDK and API versions			
Payment, success, and failure return URLs			
Webhook URL and subscribed events			
Lookup, cancellation, and settlement rights			

Do not enter the key values themselves in the environment sheet. Store server secrets so they are not exposed in client code or the repository, and assign an owner to manage them. Validate the app's move to an external payment window and back, and browser switching, as a separate environment.

> **Worked example**
>
> Card testing goes first, but virtual accounts are run only after confirming that the merchant's own test keys and deposit events are ready. For methods still under review, the time to show them on the production screen is set separately.

Let the server decide amounts and states

Do not mark an order as paid just because the payment window closed on screen or the customer returned to the success URL. Match the order number, amount, and currency stored on the server against the authorization request, and apply the provider's confirmed result to the order state.

Order and payment state transition sheet

Current state	Trigger	Next state and handling	Basis for confirmation
Worked example: payment requested	Authentication complete; amount validated on the server	Update the order after the authorization result is confirmed	Order ID and provider transaction ID
Worked example: awaiting deposit	Virtual account deposit confirmed	Payment complete; fulfillment starts	Lookup and deposit event

Design item	Rule to define	Situation to validate
Final amount	Calculate product price, discounts, and points on the server	Amount changed on screen; expired coupon
Transaction IDs	Link order, payment, and cancellation IDs	Multiple payment attempts per order
Unconfirmed state	Conditions for lookup, hold, and alerts	Delayed response; dropped connection
Partial cancellation	Manage the original transaction, cumulative cancellations, and remaining balance	Repeated partial cancellations; exceeding the balance

For a virtual account, issuance and completed deposit are different states. Keep payment, refund, and shipping or entitlement states distinct as well. If you record the identifiers of the requests and events that caused each state transition, the same transaction can also be found from a customer inquiry.

Handle retries and duplicate webhooks

A payment request can be processed while only its response is lost. In that case, check the original transaction before authorizing again as a new transaction. An idempotency key identifies the same operation; set the APIs it applies to and its retention conditions according to the provider's API contract.

Retry and receipt handling rule sheet

Situation	Identifier and duplicate criteria	Handling and repeat lookup rule	Close and alert conditions
Authorization response lost			
Cancellation request resent			
Duplicate webhook received			
Late status notification			
Processing failure and repeated failure			

Verify webhooks with the authentication and verification method set out in the provider's documentation, and keep the receipt record separate from the business processing state. Process them so that a repeated event does not duplicate order fulfillment or a refund. Do not rely only on the order of status notifications; look up and confirm any transaction whose states conflict.

> **Worked example**
>
> If a cancellation response was cut off, check the existing processing state with the identifier of the same cancellation operation. A webhook that was received but failed to update internal records is logged as awaiting processing. Close it after matching the completed reprocessing against the provider's final state.

Validate normal and exception paths together

Confirm the baseline behavior with one successful payment, then widen to failure and cancellation paths. Set the applicable scenarios by payment method and by web and app environment. For each result, record whether the order state, the customer screen, and the server record say the same thing.

Pre-launch payment scenario sheet

Scenario	Expected result	Run & evidence	Defect and owner
Successful payment and order update	Authorized on the server; amount and state match		
Abandoned authentication; checkout drop-off	Treated as unpaid; retry path offered		
Amount mismatch; duplicate request	No wrong authorization or duplicate fulfillment		
Lost response; webhook resent	Recovered by lookup; no duplicate processing		
Partial and full cancellation	Cumulative cancellations and balance match		
Virtual account deposit and expiry	Order state kept distinct before and after deposit		

> **Worked example**
>
> Reproduce, in the test environment, a connection that drops before the authorization response arrives. Check that the existing transaction is looked up before the customer is asked to pay again. If the original transaction was authorized, the goods or entitlement must be provided only once.

Mark a scenario the payment method's policy does not allow as Not applicable and record why. Mark an item that could not run because of an environment problem as Blocked and one that was not run as Not run, and link them to the launch conditions.

Make sure operations staff can find transactions

A payment launch also includes lookup screens and contact paths. Support must be able to find the authorization, notification, and cancellation history by order number, and developers to check the logs with the same identifier. Assign refund execution and rule change rights separately from lookup rights.

Payment operations handover sheet

Handover item	Location and access rights	Receiving owner	Check result
Combined order and payment lookup			
Alerts for errors and unconfirmed transactions			
Refund request, approval, and execution procedure			
Filing with the provider and emergency contact			
Settlement file and field definitions			
Launch and rollback plan			

After production activation, use the agreed validation transactions to confirm that authorization and cancellation results connect to the settlement data. Put the executing owner, the approval, and the scope of checks in the launch plan. Do not mark production payment methods and settlement data as ready on test results alone.

> **What to record for the launch decision**
>
> Connect the production payment methods and key settings, scenario results, incomplete items, and the owners for transaction lookup, refunds, and settlement in one record. If conditions remain, name an owner to complete them and the time of the next check.

Publication and use

Based on your contracted payment methods and current order structure, we define the authorization, cancellation, and refund states, the exception handling, and the launch validation scope.

IXC Corp.

CEO Nam Kichul | Business registration no. 109-86-56732
4F 408, Chuncheon ICT Venture Center
7 Huseok-ro 462beon-gil, Chuncheon-si, Gangwon-do, Republic of Korea
+82-33-242-0210 | info@ixc.co.kr

www.ixc.co.kr

www.ixc.co.kr/en/services/payment-integration

www.ixc.co.kr/en/assets/payment-integration-checklist

References

Toss Payments: integrating the order-form payment widget (in Korean)

https://docs.tosspayments.com/guides/v2/payment-widget/integration

Toss Payments Core API: lookup, cancellation, and idempotency keys (in Korean)

https://docs.tosspayments.com/reference

Toss Payments: connecting webhooks (in Korean)

https://docs.tosspayments.com/guides/v2/webhook

IXC Payment Integration

https://www.ixc.co.kr/en/services/payment-integration