



Extension Points

Punchout Documentation

Guide for Magento administrators, technical project owners, and implementation teams.

LAST UPDATED
2026-09-26

SOURCE
[Open this article online](#)

EXTENSION
[Magento 2 Extension Punchout](#)

Extension Points

Keep the PunchOut package unchanged. Add project-specific behavior in a separate Magento module through dependency injection, plugins, service contracts and events. This keeps Composer updates predictable and separates customer-specific integration code from the product.

Public service contracts

Use these interfaces for the main integration workflows:

Interface	Purpose
<code>MageB2B\Punchout\Api\ContextManagerInterface</code>	Creates and reads PunchOut session contexts.
<code>MageB2B\Punchout\Api\PunchoutContextInterface</code>	Defines the active session context passed between PunchOut services and events.
<code>MageB2B\Punchout\Api\TransferServiceInterface</code>	Prepares the cart return for OCI or cXML.
<code>MageB2B\Punchout\Api\ProtocolAdapterInterface</code>	Defines protocol-specific cart output.
<code>MageB2B\Punchout\Api\InboundOrderManagementInterface</code>	Processes and manages inbound cXML orders.
<code>MageB2B\Punchout\Api\InboundOrderRepositoryInterface</code>	Reads durable inbound-order records.
<code>MageB2B\Punchout\Api\JitCustomerProvisionerInterface</code>	Creates customers for JIT e-mail mapping.
<code>MageB2B\Punchout\Api\LineConfirmationManagementInterface</code>	Validates line decisions and queues one manual cXML confirmation.
<code>MageB2B\Punchout\Api\OciProductIdentifierInterface</code>	Resolves the configured OCI external product identifier.

Interface	Purpose
<code>MageB2B\Punchout\Api\OrderLineMetadataResolverInterface</code>	Resolves external line numbers and units for outbound documents.
<code>MageB2B\Punchout\Api\CxmlBuyerCredentialResolverInterface</code>	Resolves the effective cXML Buyer credentials for durable outbound documents.
<code>MageB2B\Punchout\Api\OutboundDocumentEndpointResolverInterface</code>	Resolves the effective Confirmation, ASN or Invoice endpoint.
<code>MageB2B\Punchout\Api\TraceRepositoryInterface</code>	Persists sanitized technical trace records.
<code>MageB2B\Punchout\Api\Bridge\ActorResolverInterface</code>	Supplies project-specific actor context.
<code>MageB2B\Punchout\Api\Bridge\PermissionGuardInterface</code>	Applies project-specific transfer permissions.

Prefer a plugin when the standard service already performs the required workflow and you only need to adjust its input or result. Use a preference only when your implementation must replace the complete contract.

Inject the interface rather than a concrete model. Magento's dependency injection configuration supplies the default implementation, and a separate integration module can add a plugin or replace the contract where a full implementation is required.

Do not use the trace repository for operational payload storage. Its input is intended for sanitized diagnostics. Inbound-order and document services retain their own operational data separately.

Lifecycle events

Event	When to use it
<code>punchout_actor_resolve_before</code>	Add actor candidates before actor resolution.

Event	When to use it
<code>punchout_actor_resolve_after</code>	Read the resolved actor context and source.
<code>punchout_permission_check_before</code>	Add project data before transfer authorization.
<code>punchout_permission_check_after</code>	React to a successful permission check.
<code>punchout_transfer_prepare_before</code>	Add data before Magento prepares the cart payload.
<code>punchout_transfer_prepare_after</code>	Inspect or extend a valid prepared transfer result.
<code>punchout_transfer_success</code>	React after Magento has generated the handoff payload. This event does not prove that the receiving browser processed it.
<code>punchout_customer_authentication</code>	React after Magento has checked customer status, lock state and confirmation.
<code>punchout_inbound_identity_resolved</code>	Adjust normalized identity data before customer mapping.
<code>punchout_jit_customer_created</code>	React after PunchOut creates a Magento customer through JIT provisioning.

Event	When to use it
<code>punchout_inbound_address_resolved</code>	Adjust a resolved billing or shipping address before quote assignment.
<code>punchout_inbound_order_processed</code>	React after inbound-order processing has reached its result.
<code>punchout_cxml_order_request_build_before_serialize</code>	Add supported cXML data to the mutable document before serialization.
<code>punchout_mapping_source_field_options</code>	Add fields to an Admin mapping source.
<code>punchout_mapping_context_values</code>	Supply runtime values for context mapping rules.
<code>punchout_document_order_after</code>	React to a Magento order document event with durable PunchOut context.
<code>punchout_document_shipment_after</code>	React to a shipment document event.
<code>punchout_document_invoice_after</code>	React to an invoice document event.
<code>punchout_document_event</code>	Handle order, shipment and invoice events through one common event.

Observers should remain small and must not modify stored credentials or unsanitized trace data. Put substantial integration logic behind an injected service and call that service from the observer.

Mapping extensions

Use `punchout_mapping_source_field_options` to expose a new typed source field in Magento Admin. Supply its runtime value through `punchout_mapping_context_values`. This keeps the mapping editor and runtime evaluation in sync.

Protected protocol fields remain owned by PunchOut. For example, the OCI external product ID comes from the buyer profile and cannot be replaced by an ordinary mapping rule.