Custom Ecommerce Integration Spec

This document describes what the custom ecommerce implementation must provide so it can be configured in the admin fields:

- "Custom Clone Endpoint"
- "Custom Categories Endpoint"

Moreover it describes:

- Autologin / SSO behavior
- Auto-post form (repost cart back to middleware)

It contains required HTTP verbs, parameters, parameter meanings and possible values, expected responses (success & error) and concrete examples.

---

1) Custom Clone Endpoint (purpose)

Used by the middleware to:

- Create a new order/session on the remote store ("create" flow), or
- Edit/populate an existing order/cart (EDIT flow) The endpoint must return either:
- An SSO URL that logs the buyer with one-use token into the remote site, where the user can be redirected to the cart if it's EDIT flow and cart is popolated, can be redirected to the selected product (in case it's communicated during the "create" flow), otherwhise is redirected to the homepage

Recommended URL path (example):

- POST https://your-store.example.com/api/punchout/clone

HTTP Verb

- POST (mandatory). Use JSON body (application/json). The middleware will call POST.

Request Body (JSON)

Base fields (always present)

- username (string) — username in the ecommerce of the user to be cloned to generate a new user bound to the punchout session.
- api_key (string) — api key that should match the one set in Punchout Rocket for the current ecommerce
- end_customer_id (integer) - end customer identifier in Punchout Rocket middleware for the ecommerce for the current punchout request
- session_token (string) — unique Buyer Cookie in case of CXML or unique UUID or similar in case of OCI
- operation (string) — "create" or "edit" or "inspect" in case of CXML, "create" in case of OCI. In case of "edit" or "inspect" a list of Cart Items might be passed

- gateway_base_url (string) — Url of the Middleware where the cart should be later posted to

Protocol-specific additions

- If the resolved protocol is CXML , these additional fields may be added:

selected_item (object) — identification of the product the user should be redirected to optional in case of "create" operation:

  - supplier_part_id (string) — identifier of the SKU (product code) in the ecommerce (if present)
  - supplier_part_auxiliary_id (string) — identifier of the product id in the ecommerce (if present) Example:
  - {
  - "selected_item": {
  - "supplier_part_id": "ABC-001",
  - "supplier_part_auxiliary_id": "19852"
  - }
  - }

cart_items (array) — the list of cart items to be readded to current session's cart Each item in cart_items typically contains:

  - sku (string) — identifier of the SKU (product code) in the ecommerce (if present)
  - product_id (string) — identifier of the product id in the ecommerce (if present)
  - quantity (integer) - quantity of the item in the cart
  - price (decimal number with dot as decimal separator) - unit price of the item in the cart. Keep in mind that it will be updated the current price when the user will be redirect to the cart
  - description (string) - description of the product
  - currency (string) — ISO 4217 currency code

Example payload (create, CXML, selected item present)


```
{
 "username": "buyer123",
 "api_key": "site-api-key-abc",
 "session_token": "sess-012345",
 "operation": "create",
 "end_customer_id" : 2,
 "gateway_base_url": "https://middleware.example.com",
 "selected_item": {
```

```
   "supplier_part_id": "X-100",
   "supplier_part_auxiliary_id": "19853"
  }
}
```


Example payload (edit, CXML, cart populated)


```
{
  "username": "buyer123",
  "api_key": "site-api-key-abc",
  "session_token": "sess-67890",
  "operation": "edit",
  "end_customer_id" : 2,
  "gateway_base_url": "https://middleware.example.com",
  "cart_items": [
   {
    "sku": "ABC-001",
    "product_id": "19852",
    "quantity": 2,
    "price": 15.95,
    "description" : "Example description",
    "currency": "EUR"
   },
   {
    "sku": "XYZ-002",
    "product_id": "19854",
    "quantity": 1,
    "price": 249.00,
    "description" : "Another example description",
    "currency": "EUR"
   }
  ]
}
```


Success Response(s)

Option A — Provide a direct SSO URL (JSON)

- HTTP 200
- Content-Type: application/json
- Body:
- {
- "status":"ok",
- "sso_url":"https://your-store.example.com/sso?token=eyJ..."
- }

Behavior: The middleware will redirect the buyer to sso_url (GET) send it back to the procurement platform in case of CXML or directly in case of OCI. The remote store must consume the token, log the user in, then:

- If action is "edit" or "inspect": the remote site should read the cart information and populate the cart so the user lands in the store in the cart page with items already in the cart matching quantities.
- If action is "create": the remote site should redirect to the selected item if passed and found, otherwise just land on the storefront

Error Responses

- HTTP 400 — Bad request (missing required fields)
- {
- "status":"error",
- "error_code":"invalid_request",
- "message":"Missing field: action"
- }
- HTTP 422 — Cannot build SSO (e.g., missing buyer identity)
- {
- "status":"error",
- "error_code":"sso_unavailable",
- "message":"Cannot generate SSO token for this user"
- }
- HTTP 500 — Internal error
- {
- "status":"error",
- "error_code":"internal_error"

- }

---

2) Autologin / SSO behavior (how to log the buyer in and populate cart)

When responding with sso_url, the store should:

- Consume the token (so it cannot be used twice) and log the buyer in with the cloned user.
- After login:
    - If action is "edit":
        - Populate the user's cart with exact products and quantities provided as cart items.
        - Redirect the user to the cart/checkout page where they can edit before checkout.
    - If action is "create":
        - Optionally redirect to the product specified as selected item. If middleware provided no select item or if it's not found in the ecommerce anymore, land the user on store home.

Security:

- Use HTTPS only.
- Tokens must be single-use.

---

3) Auto-post form (repost cart back to middleware)

When the store must send cart/cart confirmation back to middleware (after SSO login adding products to the cart or editing) when the user click the button (normally in the cart) that should initiate the cart checkout and instead in case of cloned users of punchout sessions it should POST a form (content-type: application/x-www-form-urlencoded) to middleware url communicated in the request to the cloning endpoint in the relative path "/start-sso-checkout". The preferred UX: the store serves an HTML page showing a loader that auto-submits a hidden form to the middleware endpoint (this makes SSO seamless).

Form fields expected by middleware:

- session_token (string) — session token received by the ecommerce during the clone request
- end_customer_id (integer) — end customer identification received by the ecommerce during the clone request
- For each item of the cart we should have a zero-index form data for example with key products[0][product_id] for the product id of the first item in cart. For each item in cart the list of fields is
    - sku (string) — identifier of the SKU (product code) in the ecommerce (if present)
    - product_id (string) — identifier of the product id in the ecommerce (if present)

- description (string) — description of the product
- quantity (integer) — quantity of the product in the cart
- price (decimal number with dot as decimal separator) — Unit price
- currency (string) — ISO 4217 currency code
- manufacturer_name (string) - product manufacturer
- category_ids (string) — list, separated by comma, of category integer id in the ecommerce of categories to which the product belong directly (leaf in the category tree structure normally)

Example HTML auto-post form (the store returns to user's browser and auto-submits it back to middleware):

```
<!-- Example auto-post form -->
<form id="punchout_repost" method="post" action="https://platform.punchoutrocket.com/start-sso-checkout">
 <input type="hidden" name="api_key" value="site-api-key-abc"/>
 <input type="hidden" name="session_token" value="session_token"/>
 <input type="hidden" name="products[0][sku]" value="ABC-001"/>
 <input type="hidden" name="products[0][product_id]" value="19852"/>
 <input type="hidden" name="products[0][description]" value="Example description"/>
 <input type="hidden" name="products[0][quantity]" value="2"/>
 <input type="hidden" name="products[0][price]" value="31.90"/>
 <input type="hidden" name="products[0][currency]" value="EUR"/>
 <input type="hidden" name="products[0][manufacturer_name]" value="3M"/>
 <input type="hidden" name="products[0][category_ids]" value="1145,3398"/>
 <input type="hidden" name="products[1][sku]" value="XYZ-002"/>
 <input type="hidden" name="products[1][product_id]" value="19854"/>
 <input type="hidden" name="products[1][description]" value="Another example description"/>
 <input type="hidden" name="products[1][quantity]" value="4"/>
 <input type="hidden" name="products[1][price]" value="10.90"/>
 <input type="hidden" name="products[1][currency]" value="EUR"/>
 <input type="hidden" name="products[1][manufacturer_name]" value="DEWALT"/>
 <input type="hidden" name="products[1][category_ids]" value="2146,3134"/>
</form>
<script>document.getElementById('punchout_repost').submit();</script>
```

Behavior:

- The browser auto-posts to middleware_callback.
- Middleware receives the cart and continues its workflow adding category codes (eg. UNSPSC) calculated matching ecommerce categories with remapped category codes.

---

4) Custom Categories Endpoint (purpose)

Used by middleware/admin to fetch categories from the remote store to allow mapping with standard category codes (eg. UNSPSC).

Recommended URL path (example):

- POST https://your-store.example.com/api/punchout/categories

HTTP Verb

- POST (mandatory). Optional query params allowed (see below).

Query parameters

- api_key (string) — api key that should match the one set in Punchout Rocket for the current ecommerce

Success Response

- HTTP 200
- Content-Type: application/json
- Body:
- {
- "status":"ok",
- "categories":[
- {"id":1345,"name":"Office Supplies","parent_id":0},
- {"id":1847,"name":"Pens","parent_id":1345},
- {"id":2345,"name":"Electronics","parent_id":0}
-]
- }

Semantics:

- id (integer) — unique identifier (normally integer id of the category in the table of the categories in the ecommerce) of the category, that later the middleware will use while sending the cart to the procurement system with remapped categories (eg. UNSPSC codice).
- name (string) — display name

- parent_id (integer) — unique identifier (normally integer id of the category in the table of the categories in the ecommerce) of the parent category of the current category. Normally "0" if it does not have a parent

Error Responses:

- HTTP 400 — invalid query
- HTTP 500 — internal error

Example request: GET /api/punchout/categories?api_key=site-api-key-abc

Example response:


```
{
  "status":"ok",
  "categories":[
    {"id":1345,"name":"Office Supplies","parent_id":0},
    {"id":1847,"name":"Pens","parent_id":1345},
    {"id":2345,"name":"Electronics","parent_id":0}
  ]
}
```


---

5) Notes & Implementation Hints

- Always use HTTPS.
- Use single-use SSO tokens.
- When posting cart lines, maintain index order (sku[], product_id[], quantity[], unit_price[], extended_price[], categories[]). Middleware will pair items by index.