



Plugin Directory

Changeset 3480315 for woocommerce-payments/trunk/includes/class-wc-payment-gateway-wcpay.php

Timestamp: 03/11/2026 03:29:25 PM (3 weeks ago)

Author: automattic

Message: Tagging version 10.6.0

File: 1 edited

woocommerce-payments/trunk/includes/class-wc-payment-gateway-wcpay.php (19 diffs)

Unmodified Added Removed

```

woocommerce-payments/trunk/includes/class-wc-payment-gateway-wcpay.php
r3454336r3480315
38 38 use WCPay\Core\Server\Request\Create_And_Confirm_Intention;
39 39 use WCPay\Core\Server\Request\Create_And_Confirm_Setup_Intention;
40 40 use WCPay\Core\Server\Request\Create_Setup_Intention;
41 41 use WCPay\Core\Server\Request\Get_Charge;
42 42 use WCPay\Core\Server\Request\Get_Intention;
...
112 113 const USER_FORMATTED_TOKENS_LIMIT = 100;
113 114
114 const PROCESS_REDIRECT_ORDER_MISMATCH_ERROR_CODE =
'upe_process_redirect_order_id_mismatched';
115 const UPE_APPEARANCE_TRANSIENT = 'wcpay_upe_appearance';
116 const UPE_ADD_PAYMENT_METHOD_APPEARANCE_TRANSIENT =
'upe_add_payment_method_appearance';
117 const WC_BLOCKS_UPE_APPEARANCE_TRANSIENT = 'wcpay_wc_blocks_upe_appearance';
118 const UPE_BNPL_PRODUCT_PAGE_APPEARANCE_TRANSIENT =
'upe_bnpl_product_page_appearance';
119 const UPE_BNPL_CLASSIC_CART_APPEARANCE_TRANSIENT =
'upe_bnpl_classic_cart_appearance';
120 const UPE_BNPL_CART_BLOCK_APPEARANCE_TRANSIENT =
'upe_bnpl_cart_block_appearance';
121 const UPE_APPEARANCE_THEME_TRANSIENT = 'wcpay_upe_appearance_theme';
122 const UPE_ADD_PAYMENT_METHOD_APPEARANCE_THEME_TRANSIENT =
'upe_add_payment_method_appearance_theme';
123 const WC_BLOCKS_UPE_APPEARANCE_THEME_TRANSIENT =
'wcpay_wc_blocks_upe_appearance_theme';
124 const UPE_BNPL_PRODUCT_PAGE_APPEARANCE_THEME_TRANSIENT =
'upe_bnpl_product_page_appearance_theme';
125 const UPE_BNPL_CLASSIC_CART_APPEARANCE_THEME_TRANSIENT =
'upe_bnpl_classic_cart_appearance_theme';
126 const UPE_BNPL_CART_BLOCK_APPEARANCE_THEME_TRANSIENT =
'upe_bnpl_cart_block_appearance_theme';
127
128 /**
129  * The locations of appearance transients.
130  */
131 const APPEARANCE_THEME_TRANSIENTS = [
132     'checkout' => [
133         'blocks' => self::WC_BLOCKS_UPE_APPEARANCE_THEME_TRANSIENT,

```

```

r3454336r3480315
134         'classic' => self::UPE_APPEARANCE_THEME_TRANSIENT,
135     ],
136     'product_page' => [
137         'blocks' => self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_THEME_TRANSIENT,
138         'classic' => self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_THEME_TRANSIENT,
139     ],
140     'cart' => [
141         'blocks' => self::UPE_BNPL_CART_BLOCK_APPEARANCE_THEME_TRANSIENT,
142         'classic' => self::UPE_BNPL_CLASSIC_CART_APPEARANCE_THEME_TRANSIENT,
143     ],
144 ];
115 const PROCESS_REDIRECT_ORDER_MISMATCH_ERROR_CODE = 'upe_process_redirect_order_id_mismatched';
145 116
146 117 /**
...
364 335     if ( $this->is_saved_cards_enabled() ) {
365 336         array_push( $this->supports, 'tokenization', 'add_payment_method' );
337     }
338
339     // enabling the custom place order button for express checkout methods (Apple Pay, Google
Pay, Amazon Pay)
340     // only when the feature is available. Other payment methods like WooPay or card will
return `false` for `is_express_checkout()`.
341     if ( property_exists( $this, 'has_custom_place_order_button' ) && $this->payment_method-
>is_express_checkout() && \WC_Payments::get_gateway()-
>is_express_checkout_in_payment_methods_enabled() ) {
342         $this->has_custom_place_order_button = true;
343         $this->has_fields = false;
344     }
345 }
...
883 861 */
884 862 public function is_available() {
863     // Express checkout methods (Apple Pay, Google Pay, Amazon Pay) are only available
864     // in the payment methods list when the feature is enabled. Otherwise, they appear
865     // as separate express checkout buttons.
866     if ( $this->payment_method->is_express_checkout() && ! is_admin() ) {
867         if ( ! WC_Payments::get_gateway()->is_express_checkout_in_payment_methods_enabled() )
{
868             return false;
869         }
870     }
871
872     return $this->check_base_availability();
873 }
874
875 /**
876  * Checks base availability without checkout-page-specific restrictions.
877  * Used by is_available_for_express_checkout() for payment methods that are
878  * only available via express checkout (e.g., Amazon Pay).
879  *
880  * @return bool
881  */
882 protected function check_base_availability() {
883     if ( ! WC_Payments::get_gateway()->is_enabled() ) {
884         return false;
...
926 924
927 925     // Disable the gateway if it should not be displayed on the checkout page.
928     $is_gateway_enabled = in_array( $this->stripe_id, $this-
>get_payment_method_ids_enabled_at_checkout(), true ) ? true : false;
926     $is_gateway_enabled = in_array( $this->stripe_id, $this-
>get_payment_method_ids_enabled_at_checkout(), true );

```

r3454336r3480315		
929	927	if (! \$is_gateway_enabled) {
930	928	return false;
...	...	
932	930	
933	931	return parent::is_available() && ! \$this->needs_setup();
	932	}
	933	
	934	/**
	935	* Checks if the gateway is available for express checkout.
	936	* This bypasses checkout-page-specific restrictions for payment methods
	937	* that are only available via express checkout buttons.
	938	*
	939	* @return bool
	940	*/
	941	public function is_available_for_express_checkout() {
	942	if (is_admin()) {
	943	// In admin context (e.g. block editor preview), skip full availability
	944	// checks. check_base_availability() includes runtime checks (HTTPS,
	945	// currency, capability status) that can fail without an active cart
	946	// or customer session. A simple enabled check is sufficient here.
	947	return WC_Payments::get_gateway()->is_enabled() && \$this->is_enabled();
	948	}
	949	
	950	return \$this->check_base_availability();
934	951	}
935	952	
...	...	
975	992	
976	993	return false;
	994	}
	995	
	996	/**
	997	* Whether express checkout methods should appear in the payment methods list
	998	* instead of as separate express buttons.
	999	*
	1000	* Requires both the dynamic checkout place order button feature flag
	1001	* and the express_checkout_in_payment_methods gateway setting.
	1002	*
	1003	* @return bool
	1004	*/
	1005	public function is_express_checkout_in_payment_methods_enabled(): bool {
	1006	return WC_Payments_Features::is_dynamic_checkout_place_order_button_enabled()
	1007	&& 'yes' === \$this->get_option('express_checkout_in_payment_methods');
977	1008	}
978	1009	
...	...	
1717	1748	
1718	1749	// For \$0 orders, we need to save the payment method using a setup intent.
1719		\$request = Create_And_Confirm_Setup_Intention::create();
1720		\$request->set_customer(\$customer_id);
1721		
1722		// Setting the credential based on what was provided.
1723	1750	\$payment_credential = \$payment_information->get_payment_method();
	1751	
	1752	// For confirmation tokens (e.g.: through the ECE), we must create an unconfirmed
	1753	`SetupIntent`
	1754	// and let the frontend confirm it with the confirmation token.
	1754	// Stripe's SetupIntent API doesn't support confirmation_token with confirm=true
		in the same way `PaymentIntent`s do.
1724	1755	if (\$payment_information->is_using_confirmation_token()) {
1725		\$request->set_confirmation_token(\$payment_credential);
	1756	\$request = Create_Setup_Intention::create();

```

r3454336r3480315
1757         $request->set_customer( $customer_id );
1758         $request->set_payment_method_types( $this->get_payment_method_types(
$payment_information ) );
1759         $request->set_metadata( $metadata );
1760         $request->assign_hook( 'wcpay_create_setup_intention_request' );
1726 1761     } else {
1762         $request = Create_And_Confirm_Setup_Intention::create();
1763         $request->set_customer( $customer_id );
1727 1764         $request->set_payment_method( $payment_credential );
1728     }
1729     $request->set_metadata( $metadata );
1730     $request->assign_hook( 'wcpay_create_and_confirm_setup_intention_request' );
1731     $request->set_hook_args( $payment_information, false, $save_user_in_woopay );
1732
1733     if (
1734         Payment_Method::CARD === $this->get_selected_stripe_payment_type_id() &&
1735         in_array( Payment_Method::LINK, $this->get_upe_enabled_payment_method_ids(),
true )
1765         $request->set_metadata( $metadata );
1766         $request->assign_hook( 'wcpay_create_and_confirm_setup_intention_request' );
1767         $request->set_hook_args( $payment_information, false, $save_user_in_woopay );
1768
1769         if (
1770             Payment_Method::CARD === $this->get_selected_stripe_payment_type_id() &&
1771             in_array( Payment_Method::LINK, $this-
>get_upe_enabled_payment_method_ids(), true )
1736 1772         ) {
1737         $request->set_payment_method_types( $this->get_payment_method_types(
$payment_information ) );
1738         $request->set_mandate_data( $this->get_mandate_data() );
1773         $request->set_payment_method_types( $this->get_payment_method_types(
$payment_information ) );
1774         $request->set_mandate_data( $this->get_mandate_data() );
1775     }
1739 1776     }
1740 1777
...
1812 1849     }
1813 1850
1814     if ( Intent_Status::REQUIRES_ACTION === $status ) {
1851         $needs_frontend_confirmation = (
1852             Intent_Status::REQUIRES_ACTION === $status
1853             || Intent_Status::REQUIRES_CONFIRMATION === $status
1854             || (
1855                 // For SetupIntents with confirmation tokens, the status will be
'requires_payment_method'
1856                 // since no payment method is attached yet (the confirmation token will be
used on frontend).
1857                 Intent_Status::REQUIRES_PAYMENT_METHOD === $status
1858                 && $payment_information->is_using_confirmation_token()
1859                 && ! $payment_needed
1860             )
1861         );
1862
1863         if ( $needs_frontend_confirmation ) {
1815 1864             $next_action_type = $next_action['type'] ?? null;
1816 1865             if ( 'redirect_to_url' === $next_action_type && ! empty( $next_action[
$next_action_type ][ 'url' ] ) ) {
...
1828 1877         );
1829 1878     } else {
1879         // Build the redirect URL with the confirmation token for `SetupIntent`s
requested through the ECE.

```

```

r3454336r3480315
1880 // Format: #wcpay-confirm-{si|pi}:{orderId}:{clientSecret}:{nonce}[:
{confirmationToken}].
1881 $redirect_hash_parts = [
1882     $payment_needed ? 'pi' : 'si',
1883     $order_id,
1884     $client_secret,
1885     wp_create_nonce( 'wcpay_update_order_status_nonce' ),
1886 ];
1887
1888 // For ECE SetupIntents, include the confirmation token so the frontend can
1889 // use it with confirmSetup() to complete the confirmation.
1890 if ( ! $payment_needed && $payment_information->is_using_confirmation_token()
) {
1891     $redirect_hash_parts[] = $payment_information->get_payment_method();
1892 }
1893
1894 $response = [
1895     'result' => 'success',
1896     // Include a new nonce for update_order_status to ensure the update order
1897     // status call works when a guest user creates an account during checkout.
1898     'redirect' => sprintf(
1899         '#wcpay-confirm-%s:%s:%s:%s',
1900         $payment_needed ? 'pi' : 'si',
1901         $order_id,
1902         $client_secret,
1903         wp_create_nonce( 'wcpay_update_order_status_nonce' ),
1904     ),
1905     'redirect' => '#wcpay-confirm-' . implode( ':', $redirect_hash_parts
),
1906     // Include the payment method ID so the Blocks integration can save cards.
1907     'payment_method' => $payment_information->get_payment_method(),
1908 ];
1909
1910 ...
1911
1912 $this->set_payment_method_title_for_order( $order, $payment_method_type,
$payment_method_details );
1913
1914
1915 if ( isset( $status ) && Intent_Status::REQUIRES_ACTION === $status && $this-
>is_changing_payment_method_for_subscription() ) {
1916     if ( isset( $status ) && ( Intent_Status::REQUIRES_ACTION === $status ||
Intent_Status::REQUIRES_CONFIRMATION === $status ) && $this-
>is_changing_payment_method_for_subscription() ) {
1917         // Because we're filtering woocommerce_subscriptions_update_payment_via_pay_shortcode,
1918         // we need to manually set this delayed update all flag here.
1919         if ( isset( $_POST['update_all_subscriptions_payment_method'] ) && wc_clean(
wp_unslash( $_POST['update_all_subscriptions_payment_method'] ) ) ) { // phpcs:ignore
WordPress.Security.NonceVerification.Missing
1920             // Otherwise, $gateway_id must be `woocommerce_payments`.
1921             if ( substr( $gateway_id, 0, strlen( $split_upe_gateway_prefix ) ) ===
$split_upe_gateway_prefix ) {
1922                 return [ str_replace( $split_upe_gateway_prefix, '', $gateway_id ) ];
1923                 $payment_method = str_replace( $split_upe_gateway_prefix, '', $gateway_id );
1924
1925                 // Apple Pay and Google Pay are wrappers around card payments for Stripe.
1926                 $card_wrappers = [ Payment_Method::APPLE_PAY, Payment_Method::GOOGLE_PAY ];
1927                 if ( in_array( $payment_method, $card_wrappers, true ) ) {
1928                     return [ Payment_Method::CARD ];
1929                 }
1930
1931                 return [ $payment_method ];
1932             }
1933         }
1934
1935     }
1936
1937     }
1938
1939     }
1940
1941     }
1942
1943     }
1944
1945     }
1946
1947     }
1948
1949     }
1950
1951     }
1952
1953     }
1954
1955     }
1956
1957     }
1958
1959     }
1960
1961     }
1962
1963     }
1964
1965     }
1966
1967     }
1968
1969     }
1970
1971     }
1972
1973     }
1974
1975     }
1976
1977     }
1978
1979     }
1980
1981     }
1982
1983     }
1984
1985     }
1986
1987     }
1988
1989     }
1990
1991     }
1992
1993     }
1994
1995     }
1996
1997     }
1998
1999     }
2000
2001     }
2002
2003     }
2004
2005     }
2006
2007     }
2008
2009     }
2010
2011     }
2012
2013     }
2014
2015     }
2016
2017     }
2018
2019     }
2020
2021     }
2022
2023     }
2024
2025     }
2026
2027     }
2028
2029     }
2030
2031     }
2032
2033     }
2034
2035     }
2036
2037     }
2038
2039     }
2040
2041     }
2042
2043     }
2044
2045     }
2046
2047     }
2048
2049     }
2050
2051     }
2052
2053     }
2054
2055     }
2056
2057     }
2058
2059     }
2060
2061     }
2062
2063     }
2064
2065     }
2066
2067     }
2068
2069     }
2070
2071     }
2072
2073     }
2074
2075     }
2076
2077     }
2078
2079     }
2080
2081     }
2082
2083     }
2084
2085     }
2086
2087     }
2088
2089     }
2090
2091     }
2092
2093     }
2094
2095     }
2096
2097     }
2098
2099     }
2100
2101     }
2102
2103     }
2104
2105     }
2106
2107     }
2108
2109     }
2110
2111     }
2112
2113     }
2114
2115     }
2116
2117     }
2118
2119     }
2120
2121     }
2122
2123     }
2124
2125     }
2126
2127     }
2128
2129     }
2130
2131     }
2132
2133     }
2134
2135     }
2136
2137     }
2138
2139     }
2140
2141     }
2142
2143     }
2144
2145     }
2146
2147     }
2148
2149     }
2150
2151     }
2152
2153     }
2154
2155     }
2156
2157     }
2158
2159     }
2160
2161     }
2162
2163     }
2164
2165     }
2166
2167     }
2168
2169     }
2170
2171     }
2172
2173     }
2174
2175     }
2176
2177     }
2178
2179     }
2180
2181     }
2182
2183     }
2184
2185     }
2186
2187     }
2188
2189     }
2190
2191     }
2192
2193     }
2194
2195     }
2196
2197     }
2198
2199     }
2200
2201     }
2202
2203     }
2204
2205     }
2206
2207     }
2208
2209     }
2210
2211     }
2212
2213     }
2214
2215     }
2216
2217     }
2218
2219     }
2220
2221     }
2222
2223     }
2224
2225     }
2226
2227     }
2228
2229     }
2229
2230     }
2231
2232     }
2233
2234     }
2235
2236     }
2237
2238     }
2239
2240     }
2241
2242     }
2243
2244     }
2245
2246     }
2246
2247     }
2247
2248     }
2248
2249     }
2249
2250     }
2250
2251     }
2251
2252     }
2252
2253     }
2253
2254     }
2254
2255     }
2255
2256     }
2256
2257     }
2257
2258     }
2258
2259     }
2259
2260     }
2260
2261     }
2261
2262     }
2262
2263     }
2263
2264     }
2264
2265     }
2265
2266     }
2266
2267     }
2267
2268     }
2268
2269     }
2269
2270     }
2270
2271     }
2271
2272     }
2272
2273     }
2273
2274     }
2274
2275     }
2275
2276     }
2276
2277     }
2277
2278     }
2278
2279     }
2279
2280     }
2280
2281     }
2281
2282     }
2282
2283     }
2283
2284     }
2284
2285     }
2285
2286     }
2286
2287     }
2287
2288     }
2288
2289     }
2289
2290     }
2290
2291     }
2291
2292     }
2292
2293     }
2293
2294     }
2294
2295     }
2295
2296     }
2296
2297     }
2297
2298     }
2298
2299     }
2299
2300     }
2300
2301     }
2301
2302     }
2302
2303     }
2303
2304     }
2304
2305     }
2305
2306     }
2306
2307     }
2307
2308     }
2308
2309     }
2309
2310     }
2310
2311     }
2311
2312     }
2312
2313     }
2313
2314     }
2314
2315     }
2315
2316     }
2316
2317     }
2317
2318     }
2318
2319     }
2319
2320     }
2320
2321     }
2321
2322     }
2322
2323     }
2323
2324     }
2324
2325     }
2325
2326     }
2326
2327     }
2327
2328     }
2328
2329     }
2329
2330     }
2330
2331     }
2331
2332     }
2332
2333     }
2333
2334     }
2334
2335     }
2335
2336     }
2336
2337     }
2337
2338     }
2338
2339     }
2339
2340     }
2340
2341     }
2341
2342     }
2342
2343     }
2343
2344     }
2344
2345     }
2345
2346     }
2346
2347     }
2347
2348     }
2348
2349     }
2349
2350     }
2350
2351     }
2351
2352     }
2352
2353     }
2353
2354     }
2354
2355     }
2355
2356     }
2356
2357     }
2357
2358     }
2358
2359     }
2359
2360     }
2360
2361     }
2361
2362     }
2362
2363     }
2363
2364     }
2364
2365     }
2365
2366     }
2366
2367     }
2367
2368     }
2368
2369     }
2369
2370     }
2370
2371     }
2371
2372     }
2372
2373     }
2373
2374     }
2374
2375     }
2375
2376     }
2376
2377     }
2377
2378     }
2378
2379     }
2379
2380     }
2380
2381     }
2381
2382     }
2382
2383     }
2383
2384     }
2384
2385     }
2385
2386     }
2386
2387     }
2387
2388     }
2388
2389     }
2389
2390     }
2390
2391     }
2391
2392     }
2392
2393     }
2393
2394     }
2394
2395     }
2395
2396     }
2396
2397     }
2397
2398     }
2398
2399     }
2399
2400     }
2400
2401     }
2401
2402     }
2402
2403     }
2403
2404     }
2404
2405     }
2405
2406     }
2406
2407     }
2407
2408     }
2408
2409     }
2409
2410     }
2410
2411     }
2411
2412     }
2412
2413     }
2413
2414     }
2414
2415     }
2415
2416     }
2416
2417     }
2417
2418     }
2418
2419     }
2419
2420     }
2420
2421     }
2421
2422     }
2422
2423     }
2423
2424     }
2424
2425     }
2425
2426     }
2426
2427     }
2427
2428     }
2428
2429     }
2429
2430     }
2430
2431     }
2431
2432     }
2432
2433     }
2433
2434     }
2434
2435     }
2435
2436     }
2436
2437     }
2437
2438     }
2438
2439     }
2439
2440     }
2440
2441     }
2441
2442     }
2442
2443     }
2443
2444     }
2444
2445     }
2445
2446     }
2446
2447     }
2447
2448     }
2448
2449     }
2449
2450     }
2450
2451     }
2451
2452     }
2452
2453     }
2453
2454     }
2454
2455     }
2455
2456     }
2456
2457     }
2457
2458     }
2458
2459     }
2459
2460     }
2460
2461     }
2461
2462     }
2462
2463     }
2463
2464     }
2464
2465     }
2465
2466     }
2466
2467     }
2467
2468     }
2468
2469     }
2469
2470     }
2470
2471     }
2471
2472     }
2472
2473     }
2473
2474     }
2474
2475     }
2475
2476     }
2476
2477     }
2477
2478     }
2478
2479     }
2479
2480     }
2480
2481     }
2481
2482     }
2482
2483     }
2483
2484     }
2484
2485     }
2485
2486     }
2486
2487     }
2487
2488     }
2488
2489     }
2489
2490     }
2490
2491     }
2491
2492     }
2492
2493     }
2493
2494     }
2494
2495     }
2495
2496     }
2496
2497     }
2497
2498     }
2498
2499     }
2499
2500     }
2500
2501     }
2501
2502     }
2502
2503     }
2503
2504     }
2504
2505     }
2505
2506     }
2506
2507     }
2507
2508     }
2508
2509     }
2509
2510     }
2510
2511     }
2511
2512     }
2512
2513     }
2513
2514     }
2514
2515     }
2515
2516     }
2516
2517     }
2517
2518     }
2518
2519     }
2519
2520     }
2520
2521     }
2521
2522     }
2522
2523     }
2523
2524     }
2524
2525     }
2525
2526     }
2526
2527     }
2527
2528     }
2528
2529     }
2529
2530     }
2530
2531     }
2531
2532     }
2532
2533     }
2533
2534     }
2534
2535     }
2535
2536     }
2536
2537     }
2537
2538     }
2538
2539     }
2539
2540     }
2540
2541     }
2541
2542     }
2542
2543     }
2543
2544     }
2544
2545     }
2545
2546     }
2546
2547     }
2547
2548     }
2548
2549     }
2549
2550     }
2550
2551     }
2551
2552     }
2552
2553     }
2553
2554     }
2554
2555     }
2555
2556     }
2556
2557     }
2557
2558     }
2558
2559     }
2559
2560     }
2560
2561     }
2561
2562     }
2562
2563     }
2563
2564     }
2564
2565     }
2565
2566     }
2566
2567     }
2567
2568     }
2568
2569     }
2569
2570     }
2570
2571     }
2571
2572     }
2572
2573     }
2573
2574     }
2574
2575     }
2575
2576     }
2576
2577     }
2577
2578     }
2578
2579     }
2579
2580     }
2580
2581     }
2581
2582     }
2582
2583     }
2583
2584     }
2584
2585     }
2585
2586     }
2586
2587     }
2587
2588     }
2588
2589     }
2589
2590     }
2590
2591     }
2591
2592     }
2592
2593     }
2593
2594     }
2594
2595     }
2595
2596     }
2596
2597     }
2597
2598     }
2598
2599     }
2599
2600     }
2600
2601     }
2601
2602     }
2602
2603     }
2603
2604     }
2604
2605     }
2605
2606     }
2606
2607     }
2607
2608     }
2608
2609     }
2609
2610     }
2610
2611     }
2611
2612     }
2612
2613     }
2613
2614     }
2614
2615     }
2615
2616     }
2616
2617     }
2617
2618     }
2618
2619     }
2619
2620     }
2620
2621     }
2621
2622     }
2622
2623     }
2623
2624     }
2624
2625     }
2625
2626     }
2626
2627     }
2627
2628     }
2628
2629     }
2629
2630     }
2630
2631     }
2631
2632     }
2632
2633     }
2633
2634     }
2634
2635     }
2635
2636     }
2636
2637     }
2637
2638     }
2638
2639     }
2639
2640     }
2640
2641     }
2641
2642     }
2642
2643     }
2643
2644     }
2644
2645     }
2645
2646     }
2646
2647     }
2647
2648     }
2648
2649     }
2649
2650     }
2650
2651     }
2651
2652     }
2652
2653     }
2653
2654     }
2654
2655     }
2655
2656     }
2656
2657     }
2657
2658     }
2658
2659     }
2659
2660     }
2660
2661     }
2661
2662     }
2662
2663     }
2663
2664     }
2664
2665     }
2665
2666     }
2666
2667     }
2667
2668     }
2668
2669     }
2669
2670     }
2670
2671     }
2671
2672     }
2672
2673     }
2673
2674     }
2674
2675     }
2675
2676     }
2676
2677     }
2677
2678     }
2678
2679     }
2679
2680     }
2680
2681     }
2681
2682     }
2682
2683     }
2683
2684     }
2684
2685     }
2685
2686     }
2686
2687     }
2687
2688     }
2688
2689     }
2689
2690     }
2690
2691     }
2691
2692     }
2692
2693     }
2693
2694     }
2694
2695     }
2695
2696     }
2696
2697     }
2697
2698     }
2698
2699     }
2699
2700     }
2700
2701     }
2701
2702     }
2702
2703     }
2703
2704     }
2704
2705     }
2705
2706     }
2706
2707     }
2707
2708     }
2708
2709     }
2709
2710     }
2710
2711     }
2711
2712     }
2712
2713     }
2713
2714     }
2714
2715     }
2715
2716     }
2716
2717     }
2717
2718     }
2718
2719     }
2719
2720     }
2720
2721     }
2721
2722     }
2722
2723     }
2723
2724     }
2724
2725     }
2725
2726     }
2726
2727     }
2727
2728     }
2728
2729     }
2729
2730     }
2730
2731     }
2731
2732     }
2732
2733     }
2733
2734     }
2734
2735     }
2735
2736     }
2736
2737     }
2737
2738     }
2738
2739     }
2739
2740     }
2740
2741     }
2741
2742     }
2742
2743     }
2743
2744     }
2744
2745     }
2745
2746     }
2746
2747     }
2747
2748     }
2748
2749     }
2749
2750     }
2750
2751     }
2751
2752     }
2752
2753     }
2753
2754     }
2754
2755     }
2755
2756     }
2756
2757     }
2757
2758     }
2758
2759     }
2759
2760     }
2760
2761     }
2761
2762     }
2762
2763     }
2763
2764     }
2764
2765     }
2765
2766     }
2766
2767     }
2767
2768     }
2768
2769     }
2769
2770     }
2770
2771     }
2771
2772     }
2772
2773     }
2773
2774     }
2774
2775     }
2775
2776     }
2776
2777     }
2777
2778     }
2778
2779     }
2779
2780     }
2780
2781     }
2781
2782     }
2782
2783     }
2783
2784     }
2784
2785     }
2785
2786     }
2786
2787     }
2787
2788     }
2788
2789     }
2789
2790     }
2790
2791     }
2791
2792     }
2792
2793     }
2793
2794     }
2794
2795     }
2795
2796     }
2796
2797     }
2797
2798     }
2798
2799     }
2799
2800     }
2800
2801     }
2801
2802     }
2802
2803     }
2803
2804     }
2804
2805     }
2805
2806     }
2806
2807     }
2807
2808     }
2808
2809     }
2809
2810     }
2810
2811     }
2811
2812     }
2812
2813     }
2813
2814     }
2814
2815     }
2815
2816     }
2816
2817     }
2817
2818     }
2818
2819     }
2819
2820     }
2820
2821     }
2821
2822     }
2822
2823     }
2823
2824     }
2824
2825     }
2825
2826     }
2826
2827     }
2827
2828     }
2828
2829     }
2829
2830     }
2830
2831     }
2831
2832     }
2832
2833     }
2833
2834     }
2834
2835     }
2835
2836     }
2836
2837     }
2837
2838     }
2838
2839     }
2839
2840     }
2840
2841     }
2841
2842     }
2842
2843     }
2843
2844     }
2844
2845     }
2845
2846     }
2846
2847     }
2847
2848     }
2848
2849     }
2849
2850     }
2850
2851     }
2851
2852     }
2852
2853     }
2853
2854     }
2854
2855     }
2855
2856     }
2856
2857     }
2857
2858     }
2858
2859     }
2859
2860     }
2860
2861     }
2861
2862     }
2862
2863     }
2863
2864     }
2864
2865     }
2865
2866     }
2866
2867     }
2867
2868     }
2868
2869     }
2869
2870     }
2870
2871     }
2871
2872     }
2872
2873     }
2873
2874     }
2874
2875     }
2875
2876     }
2876
2877     }
2877
2878     }
2878
2879     }
2879
2880     }
2880
2881     }
2881
2882     }
2882
2883     }
2883
2884     }
2884
2885     }
2885
2886     }
2886
2887     }
2887
2888     }
2888
2889     }
2889
2890     }
2890
2891     }
2891
2892     }
2892
2893     }
2893
2894     }
2894
2895     }
2895
2896     }
2896
2897     }
2897
2898     }
2898
2899     }
2899
2900     }
2900
2901     }
2901
2902     }
2902
2903     }
2903
2904     }
2904
2905     }
2905
2906     }
2906
2907     }
2907
2908     }
2908
2909     }
2909
2910     }
2910
2911     }
2911
2912     }
2912
2913     }
2913
2914     }
2914
2915     }
2915
2916     }
2916
2917     }
2917
2918     }
2918
2919     }
2919
2920     }
2920
2921     }
2921
2922     }
2922
2923     }
2923
2924     }
2924
2925     }
2925
2926     }
2926
2927     }
2927
2928     }
2928
2929     }
2929
2930     }
2930
2931     }
2931
2932     }
2932
2933     }
2933
2934     }
2934
2935     }
2935
2936     }
2936
2937     }
2937
2938     }
2938
2939     }
2939
2940     }
2940
2941     }
2941
2942     }
2942
2943     }
2943
2944     }
2944
2945     }
2945
2946     }
2946
2947     }
2947
2948     }
2948
2949     }
2949
2950     }
2950
2951     }
2951
2952     }
2952
2953     }
2953
2954     }
2954
2955     }
2955
2956     }
2956
2957     }
2957
2958     }
2958
2959     }
2959
2960     }
2960
2961     }
2961
2962     }
2962
2963     }
2963
2964     }
2964
2965     }
2965
2966     }
2966
2967     }
2967
2968     }
2968
2969     }
2969
2970     }
2970
2971     }
2971
2972     }
2972
2973     }
2973
2974     }
2974
2975     }
2975
2976     }
2976
2977     }
2977
2978     }
2978
2979     }
2979
2980     }
2980
2981     }
2981
2982     }
2982
2983     }
2983
2984     }
2984
2985     }
2985
2986     }
2986
2987     }
2987
2988     }
2988
2989     }
2989
2990     }
2990
2991     }
2991
2992     }
2992
2993     }
2993
2994     }
2994
2995     }
2995
2996     }
2996
2997     }
2997
2998     }
2998
2999     }
2999
3000     }
3000
3001     }
3001
3002     }
3002
3003     }
3003
3004     }
3004
3005     }
3005
3006     }
3006
3007     }
3007
3008     }
3008
3009     }
3009
3010     }
3010
3011     }
3011
3012     }
3012
3013     }
3013
3014     }
3014
3015     }
3015
3016     }
3016
3017     }
3017
3018     }
3018
3019     }
3019
3020     }
3020
3021     }
3021
3022     }
3022
3023     }
3023
3024     }
3024
3025     }
3025
3026     }
3026
3027     }
3027
3028     }
3028
3029     }
30
```

r3454336r3480315

```

2675         $this->enabled = ! empty( $this->settings[ static::METHOD_ENABLED_KEY ] ) && 'yes' ===
$this->settings[ static::METHOD_ENABLED_KEY ] ? 'yes' : 'no';
2741
2742         // Get the basic enabled value from settings.
2743         $is_enabled = ! empty( $this->settings[ static::METHOD_ENABLED_KEY ] ) && 'yes' === $this-
>settings[ static::METHOD_ENABLED_KEY ];
2744
2745         // Card and express checkout methods are not in the UPE enabled list,
2746         // so they only need the basic enabled setting check. Without this
2747         // early return, they would fall through to the UPE list verification
2748         // below and always end up disabled.
2749         if ( 'card' === $this->stripe_id || $this->payment_method->is_express_checkout() ) {
2750             return;
2751         }
2752
2753         // For split gateways, also verify the method is in the UPE enabled list.
2754         // This prevents sync issues where a gateway has enabled=yes but isn't
2755         // actually configured for checkout in the UPE settings.
2756         if ( $is_enabled ) {
2757             $upe_enabled_methods = $this->get_upe_enabled_payment_method_ids();
2758             $this->enabled         = in_array( $this->stripe_id, $upe_enabled_methods, true ) ?
'yes' : 'no';
2759         } else {
2760             $this->enabled = 'no';
2761         }
2762     }
2763
2764     ...
2765
3683     $setup_intent_request = Get_Setup_Intention::create( $intent_id );
3684     /** @var WC_Payments_API_Setup_Intention $setup_intent */ // phpcs:ignore
Generic.Commenting.DocComment.MissingShort
3685     $intent      = $setup_intent_request->send();
3686     $status      = $intent->get_status();
3687     $charge_id   = '';
3771     $intent = $setup_intent_request->send();
3772     $status = $intent->get_status();
3773
3774     // For $0 orders (free trials), directly complete the order when SetupIntent
succeeds.
3775     // This is similar to how WC Stripe Gateway handles it - calling
payment_complete()
3776     // directly ensures the order transitions to the correct status and activates
subscriptions.
3777     // Otherwise, the order would be in a "Pending payment" state and the subscription
would be "Pending".
3778     if ( Intent_Status::SUCCEEDED === $status && ! $order->is_paid() ) {
3779         $order->payment_complete( $intent_id );
3780
3781         // Add a success note similar to mark_payment_completed().
3782         $note = sprintf(
3783             /* translators: %1: the successfully charged amount, %2: WooPayments, %3:
transaction ID of the payment */
3784             __( 'A payment of %1$s was successfully charged using %2$s (%3$s).',
'woocommerce-payments' ),
3785             wc_price( $order->get_total(), [ 'currency' => $order->get_currency() ] ),
3786             'WooPayments',
3787             $intent_id
3788         );
3789         $order->add_order_note( $note );
3790         $this->order_service->set_intention_status_for_order( $order, $status );
3791         $order->save();
3792     }
3793 }
3688 }
3689 }

```

```

r3454336r3480315
3690 3795 $payment_method_id = $intent->get_payment_method_id();
3796
3797 // For SetupIntents confirmed via frontend (e.g., ECE with confirmation tokens),
3798 // store the payment method ID in order meta. This ensures subscription renewals
3799 // can find the payment method even if token creation fails later.
3800 if ( ! empty( $payment_method_id ) ) {
3801     $this->order_service->set_payment_method_id_for_order( $order, $payment_method_id
3802 );
3803 }
3691 3803
3692 3804 if ( Intent_Status::SUCCEEDED === $status ) {
...
3711 3823     }
3712 3824     } catch ( Exception $e ) {
3713 // If saving the token fails, log the error message but catch the error to
avoid crashing the checkout flow.
3714 3825     Logger::log( 'Error when saving payment method: ' . $e->getMessage() );
3826
3827 // For subscription orders, token creation failure is critical - renewals
will fail.
3828 // Re-throw the exception so the customer sees an error instead of a
successful
3829 // checkout that will fail on the first renewal.
3830 if ( $is_subscription ) {
3831     throw new Exception(
3832         __( 'Unable to save payment method for subscription. Please try
again or use a different payment method.', 'woocommerce-payments' )
3833     );
3834 }
3715 3835     }
3716 3836 }
...
3966 4086 * Returns a formatted token list for a user.
3967 4087 *
3968 * @param int $user_id The user ID.
3969 */
3970 protected function get_user_formatted_tokens_array( $user_id ) {
4088 * @param int $user_id The user ID.
4089 * @param string|null $gateway_id Optional gateway ID to filter tokens. Defaults to card
gateway.
4090 */
4091 protected function get_user_formatted_tokens_array( $user_id, $gateway_id = null ) {
3971 4092     $tokens = WC_Payment_Tokens::get_tokens(
3972 4093     [
3973 4094         'user_id' => $user_id,
3974 'gateway_id' => self::GATEWAY_ID,
4095 'gateway_id' => $gateway_id ?? self::GATEWAY_ID,
3975 4096         'limit' => self::USER_FORMATTED_TOKENS_LIMIT,
3976 4097     ]
...
3979 4100     return array_map(
3980 4101         static function ( WC_Payment_Token $token ): array {
4102 // ensures that Google Pay/Apple Pay methods display "Google Pay Visa ending in
1234",
4103 // instead of just "Visa ending in 1234".
4104 $wallet_type = $token->get_meta( '_wcpay_wallet_type', true );
4105 $name = $token->get_display_name();
4106 $payment_method = WC_Payments::get_payment_method_by_id( $wallet_type );
4107 if ( $payment_method && method_exists( $payment_method, 'get_title' ) ) {
4108     $name = join( ' ', [ $payment_method->get_title(), $name ] );
4109 }
4110
3981 4111     return [

```

```

r3454336r3480315
3982 4112         'tokenId'         => $token->get_id(),
3983 4113         'paymentMethodId' => $token->get_token(),
3984 4114         'isDefault'     => $token->get_is_default(),
3985 4115         'displayName'   => $token->get_display_name(),
4115 4115         'displayName'   => $name,
3986 4116     ];
3987 4117     },
...
4216 4346     }
4217 4347
4218 /**
4219  * Handle AJAX request for saving UPE appearance value to transient.
4220  *
4221  * @throws Exception - If nonce or setup intent is invalid.
4222  */
4223 public function save_upe_appearance_ajax() {
4224     try {
4225         $is_nonce_valid = check_ajax_referer( 'wcpay_save_upe_appearance_nonce', false, false
);
4226         if ( ! $is_nonce_valid ) {
4227             throw new Exception(
4228                 __( 'Unable to update UPE appearance values at this time.', 'woocommerce-
payments' )
4229             );
4230         }
4231
4232         $elements_location = isset( $_POST['elements_location'] ) ? wc_clean( wp_unslash(
$_POST['elements_location'] ) ) : null;
4233         $appearance        = isset( $_POST['appearance'] ) ? json_decode( wc_clean(
wp_unslash( $_POST['appearance'] ) ) ) : null;
4234
4235         $valid_locations = [ 'blocks_checkout', 'shortcode_checkout', 'bnpl_product_page',
'bnpl_classic_cart', 'bnpl_cart_block', 'add_payment_method' ];
4236         if ( ! $elements_location || ! in_array( $elements_location, $valid_locations, true )
) {
4237             throw new Exception(
4238                 __( 'Unable to update UPE appearance values at this time.', 'woocommerce-
payments' )
4239             );
4240         }
4241
4242         if ( in_array( $elements_location, [ 'blocks_checkout', 'shortcode_checkout' ], true )
) {
4243             $is_blocks_checkout = 'blocks_checkout' === $elements_location;
4244             /**
4245              * This filter is only called on "save" of the appearance, to avoid calling it on
every page load.
4246              * If you apply changes through this filter, you'll need to clear the transient
data to see them at checkout.
4247              *
4248              * @deprecated 7.4.0 Use {@see 'wcpay_elements_appearance'} instead.
4249              * @since 7.3.0
4250              */
4251             $appearance = apply_filters_deprecated( 'wcpay_upe_appearance', [ $appearance,
$is_blocks_checkout ], '7.4.0', 'wcpay_elements_appearance' );
4252         }
4253
4254         /**
4255          * This filter is only called on "save" of the appearance, to avoid calling it on
every page load.
4256          * If you apply changes through this filter, you'll need to clear the transient data
to see them at checkout.
4257          * $elements_location can be 'blocks_checkout', 'shortcode_checkout',
'bnpl_product_page', 'bnpl_classic_cart', 'bnpl_cart_block', 'add_payment_method'.
4258          */

```

r3454336r3480315

```

4259         * @since 7.4.0
4260         */
4261         $appearance = apply_filters( 'wcpay_elements_appearance', $appearance,
$elements_location );
4262
4263         $appearance_transient = [
4264             'shortcode_checkout' => self::UPE_APPEARANCE_TRANSIENT,
4265             'add_payment_method' => self::UPE_ADD_PAYMENT_METHOD_APPEARANCE_TRANSIENT,
4266             'blocks_checkout' => self::WC_BLOCKS_UPE_APPEARANCE_TRANSIENT,
4267             'bnpl_product_page' => self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_TRANSIENT,
4268             'bnpl_classic_cart' => self::UPE_BNPL_CLASSIC_CART_APPEARANCE_TRANSIENT,
4269             'bnpl_cart_block' => self::UPE_BNPL_CART_BLOCK_APPEARANCE_TRANSIENT,
4270         ][ $elements_location ];
4271         $appearance_theme_transient = [
4272             'shortcode_checkout' => self::UPE_APPEARANCE_THEME_TRANSIENT,
4273             'add_payment_method' => self::UPE_ADD_PAYMENT_METHOD_APPEARANCE_THEME_TRANSIENT,
4274             'blocks_checkout' => self::WC_BLOCKS_UPE_APPEARANCE_THEME_TRANSIENT,
4275             'bnpl_product_page' => self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_THEME_TRANSIENT,
4276             'bnpl_classic_cart' => self::UPE_BNPL_CLASSIC_CART_APPEARANCE_THEME_TRANSIENT,
4277             'bnpl_cart_block' => self::UPE_BNPL_CART_BLOCK_APPEARANCE_THEME_TRANSIENT,
4278         ][ $elements_location ];
4279
4280         if ( null !== $appearance ) {
4281             set_transient( $appearance_transient, $appearance, DAY_IN_SECONDS );
4282             set_transient( $appearance_theme_transient, $appearance->theme, DAY_IN_SECONDS );
4283         }
4284
4285         wp_send_json_success( $appearance, 200 );
4286     } catch ( Exception $e ) {
4287         // Send back error so it can be displayed to the customer.
4288         wp_send_json_error(
4289             [
4290                 'error' => [
4291                     'message' => WC_Payments_Utils::get_filtered_error_message( $e ),
4292                 ],
4293             ],
4294             WC_Payments_Utils::get_filtered_error_status_code( $e )
4295         );
4296     }
4297 }
4298
4299 /**
4300  * Clear the saved UPE appearance transient value.
4301  */
4302 public function clear_upe_appearance_transient() {
4303     delete_transient( self::UPE_APPEARANCE_TRANSIENT );
4304     delete_transient( self::WC_BLOCKS_UPE_APPEARANCE_TRANSIENT );
4305     delete_transient( self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_TRANSIENT );
4306     delete_transient( self::UPE_BNPL_CLASSIC_CART_APPEARANCE_TRANSIENT );
4307     delete_transient( self::UPE_BNPL_CART_BLOCK_APPEARANCE_TRANSIENT );
4308     delete_transient( self::UPE_APPEARANCE_THEME_TRANSIENT );
4309     delete_transient( self::WC_BLOCKS_UPE_APPEARANCE_THEME_TRANSIENT );
4310     delete_transient( self::UPE_BNPL_PRODUCT_PAGE_APPEARANCE_THEME_TRANSIENT );
4311     delete_transient( self::UPE_BNPL_CLASSIC_CART_APPEARANCE_THEME_TRANSIENT );
4312     delete_transient( self::UPE_BNPL_CART_BLOCK_APPEARANCE_THEME_TRANSIENT );
4313 }
4348
4349
4350 /**
...
4487 4522

```

r3454336r3480315		
4488	4523	/**
4489		* Checks if UPE appearance theme is set and returns appropriate icon URL.
	4524	* Returns the appropriate icon URL for the payment method.
4490	4525	*
4491	4526	* @return string
4492	4527	*/
4493	4528	public function get_theme_icon() {
4494		\$upe_appearance_theme = get_transient(self::UPE_APPEARANCE_THEME_TRANSIENT);
4495		if (\$upe_appearance_theme) {
4496		return 'night' === \$upe_appearance_theme ? \$this->payment_method->get_dark_icon() :
4497		\$this->payment_method->get_icon();
4497		}
4498	4529	return \$this->payment_method->get_icon();
4499	4530	}

[About](#)[Showcase](#)[Learn](#)[Get Involved](#)[News](#)[Themes](#)[Documentation](#)[Events](#)[Hosting](#)[Plugins](#)[Developers](#)[Donate ↗](#)[Privacy](#)[Patterns](#)[WordPress.tv ↗](#)[Five for the Future](#)[WordPress.com ↗](#)[Matt ↗](#)[bbPress ↗](#)[BuddyPress ↗](#)

CODE IS POETRY

The WordPress® trademark is the intellectual property of the WordPress Foundation.