

 martinblech / xmltodict Public

[Code](#) [Issues](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Wiki](#) [Security and privacy](#)

# Commit f98c90f



 martinblech committed on Sep 8, 2025 · ✓ 6 / 6

Enhance unparse() XML name validation with stricter rules and tests

Extend existing validation (previously only for "<" and ">") to also reject element, attribute, and xmlns prefix names that are non-string, start with "?" or "!", or contain "/", spaces, tabs, or newlines. Update \_emit and namespace handling to use \_validate\_name. Add tests covering these new invalid name cases.

 [master](#) ·  v1.0.4 ... v0.15.1


1 parent [f759b13](#) commit f98c90f 

2 files changed

+99 -9 ■■■■■

 [↑ Top](#) 

▼  tests

 test\_dicttoxml.py

 xmltodict.py





▼ tests/test\_dicttoxml.py ...

```

@@ -263,3 +263,63 @@ def
test_attribute_values_with_angle_brackets_are_escaped(self):
263 263         xml = unparse({"a": {"@attr": "1<middle>2", "#text": "x"}},
                full_document=False)
264 264         # The generated XML should contain escaped '<' and '>' within the
                attribute value
265 265         self.assertIn('attr="1&lt;middle&gt;2"', xml)

```

```
266 +
267 +     def test_rejects_tag_name_starting_with_question(self):
268 +         with self.assertRaises(ValueError):
269 +             unparse({"?pi": "data"}, full_document=False)
270 +
271 +     def test_rejects_tag_name_starting_with_bang(self):
272 +         with self.assertRaises(ValueError):
273 +             unparse({"!decl": "data"}, full_document=False)
274 +
275 +     def test_rejects_attribute_name_starting_with_question(self):
276 +         with self.assertRaises(ValueError):
277 +             unparse({"a": {""@?weird": "x"}}, full_document=False)
278 +
279 +     def test_rejects_attribute_name_starting_with_bang(self):
280 +         with self.assertRaises(ValueError):
281 +             unparse({"a": {""@!weird": "x"}}, full_document=False)
282 +
283 +     def test_rejects_xmlns_prefix_starting_with_question_or_bang(self):
284 +         with self.assertRaises(ValueError):
285 +             unparse({"a": {"@xmlns": {"?p": "http://e/"}}},
286 + full_document=False)
287 +             unparse({"a": {"@xmlns": {"!p": "http://e/"}}},
288 + full_document=False)
289 +
290 +     def test_rejects_non_string_names(self):
291 +         class Weird:
292 +             def __str__(self):
293 +                 return "bad>name"
294 +
295 +         # Non-string element key
296 +         with self.assertRaises(ValueError):
297 +             unparse({Weird(): "x"}, full_document=False)
298 +
299 +         # Non-string attribute key
300 +         with self.assertRaises(ValueError):
301 +             unparse({"a": {Weird(): "x"}}, full_document=False)
302 +
303 +     def test_rejects_tag_name_with_slash(self):
304 +         with self.assertRaises(ValueError):
305 +             unparse({"bad/name": "x"}, full_document=False)
```

```

304 +
305 +     def test_rejects_tag_name_with_whitespace(self):
306 +         for name in ["bad name", "bad\tname", "bad\nname"]:
307 +             with self.assertRaises(ValueError):
308 +                 unparse({name: "x"}, full_document=False)
309 +
310 +     def test_rejects_attribute_name_with_slash(self):
311 +         with self.assertRaises(ValueError):
312 +             unparse({"a": {"@bad/name": "x"}}, full_document=False)
313 +
314 +     def test_rejects_attribute_name_with_whitespace(self):
315 +         for name in ["@bad name", "@bad\tname", "@bad\nname"]:
316 +             with self.assertRaises(ValueError):
317 +                 unparse({"a": {name: "x"}}, full_document=False)
318 +
319 +     def test_rejects_xmlns_prefix_with_slash_or_whitespace(self):
320 +         # Slash
321 +         with self.assertRaises(ValueError):
322 +             unparse({"a": {"@xmlns": {"bad/prefix": "http://e/"}},
323 + full_document=False)
324 +         # Whitespace
325 +         with self.assertRaises(ValueError):
326 +             unparse({"a": {"@xmlns": {"bad prefix": "http://e/"}},
327 + full_document=False)

```

▼ xmltodict.py

...

↑

@@ -368,7 +368,42 @@ def \_has\_angle\_brackets(value):

368 368 return isinstance(value, str) and ("<" in value or ">" in value)

369 369

370 370

```

371 + def _has_invalid_name_chars(value):
372 +     """Return True if value (a str) contains any disallowed name characters.
373 +
374 +     Disallowed: '<', '>', '/', or any whitespace character.
375 +     Non-string values return False.
376 +     """
377 +     if not isinstance(value, str):
378 +         return False
379 +     if "<" in value or ">" in value or "/" in value:
380 +         return True

```

```

381 + # Check for any whitespace (spaces, tabs, newlines, etc.)
382 + return any(ch.isspace() for ch in value)
383 +
384 +
385 + def _validate_name(value, kind):
386 +     """Validate an element/attribute name for XML safety.
387 +
388 +     Raises ValueError with a specific reason when invalid.
389 +
390 +     kind: 'element' or 'attribute' (used in error messages)
391 +     """
392 +     if not isinstance(value, str):
393 +         raise ValueError(f"{kind} name must be a string")
394 +     if value.startswith("?") or value.startswith("!"):
395 +         raise ValueError(f'Invalid {kind} name: cannot start with "?" or "!')
396 +     if "<" in value or ">" in value:
397 +         raise ValueError(f'Invalid {kind} name: "<" or ">" not allowed')
398 +     if "/" in value:
399 +         raise ValueError(f'Invalid {kind} name: "/" not allowed')
400 +     if any(ch.isspace() for ch in value):
401 +         raise ValueError(f"Invalid {kind} name: whitespace not allowed")
402 +
403 +

```

```

371 404 def _process_namespace(name, namespaces, ns_sep=':', attr_prefix='@'):

```

```

405 +     if not isinstance(name, str):
406 +         return name

```

```

372 407         if not namespaces:

```

```

373 408             return name

```

```

374 409         try:

```



```

@@ -402,8 +437,7 @@ def _emit(key, value, content_handler,

```

```

402 437             return

```

```

403 438                 key, value = result

```

```

404 439                 # Minimal validation to avoid breaking out of tag context

```

```

405 -                 if _has_angle_brackets(key):

```

```

406 -                     raise ValueError('Invalid element name: "<" or ">" not allowed')

```

```

440 +                 _validate_name(key, "element")

```

```

407 441                 if not hasattr(value, '__iter__') or isinstance(value, (str, dict)):

```

```

408 442                     value = [value]

```

```

409 443                 for index, v in enumerate(value):

```

		@@ -427,23 +461,19 @@
		def _emit(key, value, content_handler,
427	461	if ik == cdata_key:
428	462	cdata = iv
429	463	continue
430	-	if ik.startswith(attr_prefix):
464	+	if isinstance(ik, str) and ik.startswith(attr_prefix):
431	465	ik = _process_namespace(ik, namespaces, namespace_separator,
432	466	attr_prefix)
433	467	if ik == '@xmlns' and isinstance(iv, dict):
434	468	for k, v in iv.items():
435	-	if _has_angle_brackets(k):
436	-	raise ValueError(
437	-	'Invalid attribute name: "<" or ">" not
		allowed'
438	-	)
469	+	_validate_name(k, "attribute")
439	470	attr = 'xmlns{}'.format(f':{k}' if k else '')
440	471	attrs[attr] = str(v)
441	472	continue
442	473	if not isinstance(iv, str):
443	474	iv = str(iv)
444	475	attr_name = ik[len(attr_prefix) :]
445	-	if _has_angle_brackets(attr_name):
446	-	raise ValueError('Invalid attribute name: "<" or ">" not
		allowed')
476	+	_validate_name(attr_name, "attribute")
447	477	attrs[attr_name] = iv
448	478	continue
449	479	children.append((ik, iv))

## Comments 0



Please [sign in](#) to comment.