PublicElementConfig

Overview

The `PublicElementConfig` class is used within FLUX task processing scripts to manage and validate element configurations. It provides methods for retrieving, comparing, and auditing configuration values across multiple formats.

Configuration Object

Attributes

The `config` object provides the following attributes:

Attribute	Type	Description
`klass`	string	Format class of the current configuration. Supported values: `txt`, `py`, `xml`, `json`, `yaml`
`section`	string	Section identifier of the current configuration
`version`	string	Version identifier of the current configuration
`required_approvement_percentage`	integer	Minimum percentage of audit tests that must pass for the configuration to be considered approved
`audit_value`	integer	Final calculated result of the audit

Methods

get_last_config()

Returns the most recent configuration for the current element.

Parameters: None

Returns: Configuration object

to_json()

Converts the current configuration to JSON format.

Parameters: None

Returns: JSON string representation of the configuration

---

get(key)

Retrieves a specific value from the configuration by key.

Parameters:

- `key` (string) – The configuration key to retrieve

Returns: Value associated with the specified key

Example:

```
domain_value = config.get('domain_value')
```

---

compare_numeric_value(key, compare_operator, value, required_for_audit)

Compares a numeric configuration value against a specified value using a comparison operator.

Parameters:

- `key` (string) – Configuration key to compare
- `compare_operator` (string) – Comparison operator: `'>'`, `'>='`, `'<'`, `'<='`, `'=='`
- `value` (integer) – Value to compare against
- `required_for_audit` (bool) – Whether this comparison is required for audit approval

Returns: Boolean result of the comparison

Example:

```
settings['audit.valid_domain'] = config.compare_numeric_value('size', '>=', 4,
required_for_audit=True)
```

---

compare_string_value(key, compare_operator, value, required_for_audit)

Compares a string configuration value against a specified value using a comparison operator.

Parameters:

- `key` (string) – Configuration key to compare
- `compare_operator` (string) – Comparison operator: `'startswith'`, `'endswith'`, `'contains'`, `'=='`
- `value` (string) – Value to compare against
- `required_for_audit` (bool) – Whether this comparison is required for audit approval

Returns: Boolean result of the comparison

Example:

```
settings['audit.valid_domain'] = config.compare_string_value('domain', 'startswith', 'https', required_for_audit=True)
```

---

check_key_matches(key_pattern, required_for_audit)

Checks whether any configuration key matches the specified pattern.

Parameters:

- `key_pattern` (string) – Pattern to match against configuration keys (supports wildcards)
- `required_for_audit` (bool) – Whether this check is required for audit approval

Returns: Boolean indicating if a match was found

Example:

```
config.check_key_matches("interface.GigabitEthernet1.*", required_for_audit=True)
```

---

check_key_occurrences(key_pattern, compare_occurrences_operator, value, required_for_audit)

Counts the number of configuration keys matching a pattern and compares the count against a specified value.

Parameters:

- `key_pattern` (string) – Pattern to match against configuration keys (supports wildcards)
- `compare_occurrences_operator` (string) – Comparison operator: `'>'`, `'>='`, `'<'`, `'<='`, `'=='`
- `value` (integer) – Number to compare the occurrence count against
- `required_for_audit` (bool) – Whether this check is required for audit approval

Returns: Boolean result of the comparison

Example:

```
config.check_key_occurrences("interface.*", ">=", 90, required_for_audit=True)
```

---

complex_compare(key_pattern, compare_str, required_for_audit)

Executes a complex comparison using OpenAI natural language processing to evaluate configuration values.

Parameters:

- `key_pattern` (string) – Pattern to match against configuration keys
- `compare_str` (string) – Natural language description of the comparison criteria
- `required_for_audit` (bool) – Whether this comparison is required for audit approval

Returns: Boolean result of the comparison

Example:

```
config.complex_compare('interface.x', 'comienza con Virtual', required_for_audit=True)
```

Complete Example

The following example demonstrates a comprehensive configuration audit script using multiple validation methods:

```
# Set configuration metadata
config.version = '1.7'
config.section = 'core'
config.required_approvement_percentage = 85

# Validate interface count
settings['audit.interfaces'] = config.check_key_occurrences("interface.*", ">=", 90,
required_for_audit=True)

# Validate domain configuration
settings['audit.valid_domain'] = config.compare_string_value('domain', 'startswith',
'zequenze', required_for_audit=True)

# Complex validations using natural language
```

```
settings['audit.virtual_interface'] = config.complex_compare('interface.x', 'comienza con Virtual', required_for_audit=True)
settings['audit.custom_hop'] = config.complex_compare('route.x', 'contiene un nexthop a la ip 172.16.254.1', required_for_audit=True)

# Validate specific routing configuration
settings['audit.route'] = config.check_key_matches('route.0.0.0.0/0', required_for_audit=True)

# Validate hostname format
settings['audit.valid_hostname'] = config.compare_string_value('hostname', 'contains', '01', required_for_audit=True)

# Check for specific interface features
settings['audit.has_logging_interface'] = config.check_key_matches("interface.*.logging", required_for_audit=True)
settings['audit.has_vrf_interface'] = config.check_key_matches("interface.*.vrf", required_for_audit=True)
settings['audit.has_gigabit_interface'] = config.check_key_matches("interface.GigabitEthernet1.*", required_for_audit=True)
settings['audit.has_gigabit_two_interface'] = config.check_key_matches("interface.GigabitEthernet2.*", required_for_audit=True)

# Store final audit result
settings['audit.result'] = config.audit_value
```

Revision #1
Created 2026-02-13 23:07:58 UTC by ipena@zequenze.com
Updated 2026-03-05 22:30:35 UTC by ipena@zequenze.com