Secure Internet and SaaS Access (ZIA)

Defining Patterns for Custom DLP Dictionaries

You can use alphanumeric patterns to configure custom dictionaries that match a wide variety of data types. For example, you can define patterns to detect data like phone numbers, driver's license numbers, or credit card numbers for specific issuers (a number of sample patterns are provided below). To learn more, see Adding Custom DLP Dictionaries.

General guidelines for patterns are as follows:

- A dictionary can contain up to 8 patterns.
- Each pattern can have a maximum of 256 bytes.
- You can use tokens (`?i` and `?-i`) for case-insensitive matches.

Syntax Requirements for Patterns

Metacharacters Supported by Zscaler

Metacharacter (POSIX ERE)	Definition	Supported by Zscaler?
.	Matches any single character	Supported
[]	Matches a single character from those given between the brackets, matches character ranges (e.g., [a–z])	Supported
[^]	Matches a single character not from those given between ^ and]	Supported
^	Matches the starting position within the string	Supported
$	Matches the ending position within the string	Supported
()	Defines a capture group	Capture groups are not supported. Parentheses can be used for bounding subexpressions.

Metacharacter (POSIX ERE)	Definition	Supported by Zscaler?
\n	References a capture group	Not Supported
*	Repeats the preceding element 0 or more times	Supported
{m,n}	Matches the preceding element at least *m* times and not more than *n* times. Requires *m* is less than or equal to *n*.	Zscaler recommends specifying a range as small as possible with *m* and *n*. Additionally, you should use the *{m,n}* construct to quantify the most specific preceding characters (e.g., *[a-z]{3,5}* instead of *.{3,100}*. If you provide a range that is too wide, the pattern might not be accepted.
?	Repeats the preceding element 0 or 1 times	Supported
+	Repeats the preceding element 1 or more times	Supported
|	Matches either the expression before or the expression after the operator (e.g., "abc|def" matches "abc" or "def")	Supported

Shorthand Character Classes Supported by Zscaler

Character class	Definition	Supported by Zscaler?
\d	Matches any single character that is a digit (i.e., [0–9])	Supported
\D	Matches any single character that is a non-digit	Supported

Character class	Definition	Supported by Zscaler?
\s	Matches any single horizontal whitespace character across a single line (i.e., a space " " or tab)	Supported
\S	Matches any single non-whitespace character	Supported
\w	Matches any single word character (i.e., ASCII characters [a–zA–Z0–9_])	Supported
\W	Matches any single non-word character	Supported
\b	Matches any single word boundary	Supported

Nested Repeats

A repeat of an expression containing a repeat (*, +, ?, or {*m*,*n*}) is supported. Examples of repeated elements are in red text below:

Syntax with Nested Repeats	Syntax without Nested Repeats
\d-[A-Z]{2}?X - *Matches "5-X" or "5-GAX"*	\d-([A-Z][A-Z])?X
(\d{3}-){1,2}\d{4} - *Matches "555-555-5555" or "555-5555"* \d{3}-(\d{3}-)?\d{4}	\d{3}-(\d\d\d-)?\d{4}

Base Tokens

Your patterns are no longer required to begin with a simple sequence of alphabetic and numeric characters of a known length called the "base token". A base token ends with an expression that is non-alphanumeric (e.g., ";"), or with the end of the pattern.

The following are examples of patterns with and without base tokens:

Example - US phone numbers:

- *Without a base token*: (1-)?\d{3}-\d{3}-\d{4} - *Starts with an optional expression (in red).*
- *Without a base token: (CA)-\|d{6} - Starts with grouping (in red).*
- *With a base token:*:

- 1-\d{3}-\d{3}-\d{4} - *Starts with a single-number token (in red).*
- \d{3}-\d{3}-\d{4} - *Starts with a three-number token (in red).*

Example - IPv4 Address in dotted-quad notation:

- *Without a Base Token*: [0-9.]{7,15} - *Mix of numeric characters and punctuation (in red).*
- *With a Base Token*: \d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3} - *Starts with 1-3 numeric characters (in red).*

Sample Patterns

Australian Business Number (ABN)

```
\d{2} \d{3} \d{3} \d{3}
or
[0-9]{2} [0-9]{3} [0-9]{3} [0-9]{3}

matches:
12 333 444 555
```

California Driver's License Number

```
[A-Z]\d{7}
or
[A-Z]\d{7}

matches:
A1234567
X7654321
```

Credit Card Numbers for Specific Issuers (for example, Wells Fargo VISA)

```
4147[- ]?18\d{2}[ -]?\d{4}[ -]?\d{4}

matches:
4147180011112222
4147-1800-1111-2222
4147 1800 1111 2222
```

Hong Kong Identity Card (HKID)

```
[A-Z]{1,2}\d{6}[0-9A]

matches:
A1234567
```



```
BF1234567
N123456A
ZX123456A
```

IPv4 Address (CIDR notation)

```
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}(\/(\d|[1-2]\d|3[0-2]))?

matches:
192.168.100.0
192.168.100.0/24
```

Tokens for case-insensitive pattern matching

```
/foo-(?i)BaR-(?-i)bAZ/

matches:
BaR (case-insensitive)
bAZ (case-sensitive)
foo-BAR-bAZ

doesn't match:
foo-BAR-baz
```

SWIFT Code

```
[A-Z]{6}[A-Z0-9]{2,5}

matches:
CALCUS6L
BCLFUS66
BIMIUS33
BBVAUS33GCI
```

Unicode Characters

```
(财务|税务)部门

matches:
财务部门
```

税务部门


```
\d{2,4}年\d{1,2}月\d{1,2}日

matches:
2023年01月30日
23年1月30日
```

United States Phone Numbers

```
[2-9]\d{2}[-. ]\d{4}

matches:
555-1212
555.1212

555 1212
```

```
[2-9]\d{2}\)?[-. ][2-9]\d{2}[-. ]\d{4}

matches:
(415) 555-1212 *
415-555-1212
415 555 1212
415.555.1212

(* matches a substring)
```

Lookaround Constructs

For custom dictionaries that use patterns with lookaround constructs (also known as zero-length assertions), you must:

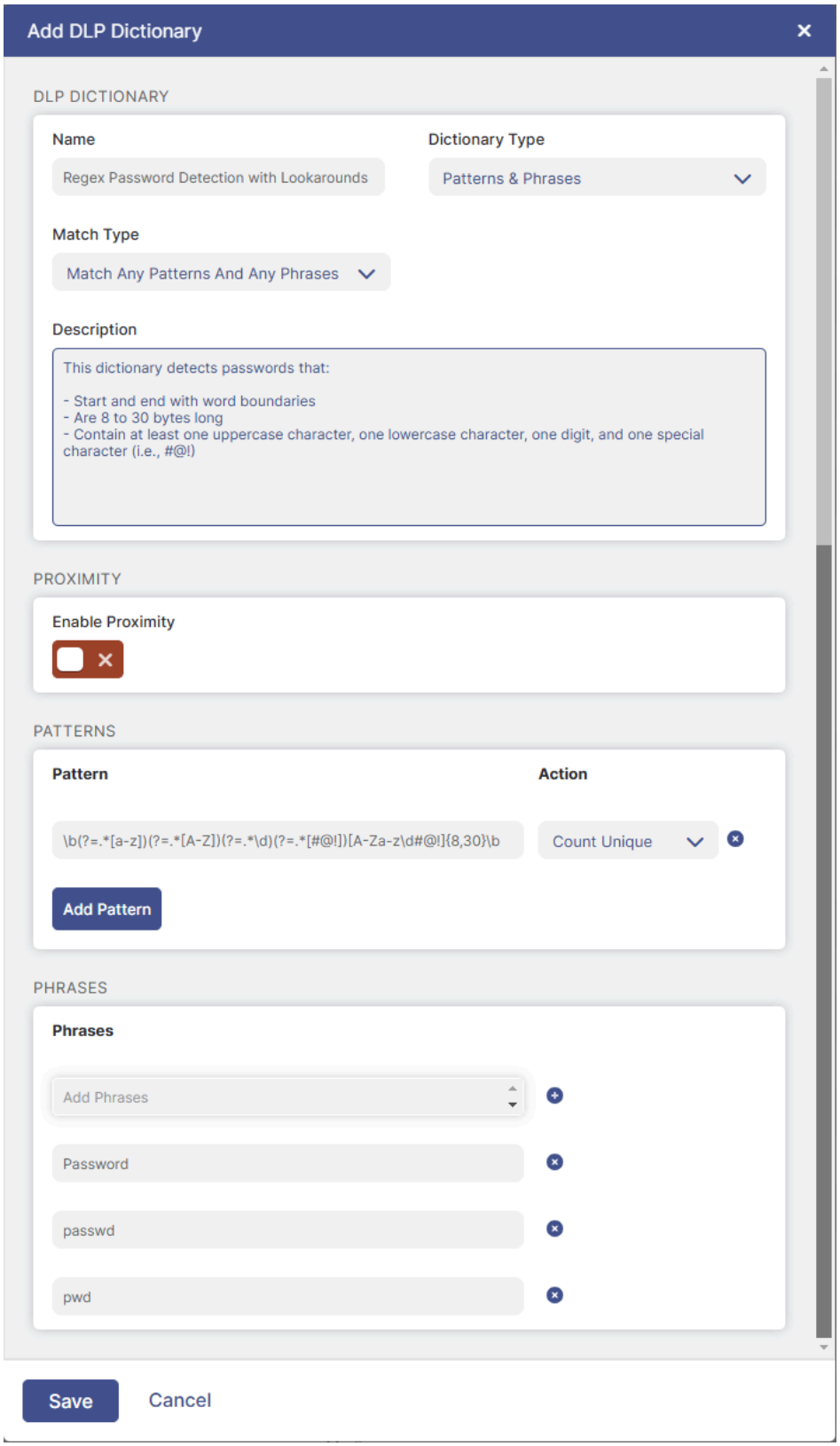
- Select **Match Any Patterns and Any Phrases** as the **Match Type**.
- Configure at least one phrase.

Consider the following examples:

Sample regex with lookaround constructs

```
\b(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[#@!])[A-Za-z\d#@!]{8,30}\b
```


Custom dictionary using sample regex



In the examples, the regex uses lookaround constructs to detect passwords that:

- Start and end with word boundaries
- Are 8 to 30 bytes long
- Contain at least one uppercase character, one lowercase character, one digit, and one special character (i.e., `#@!`)

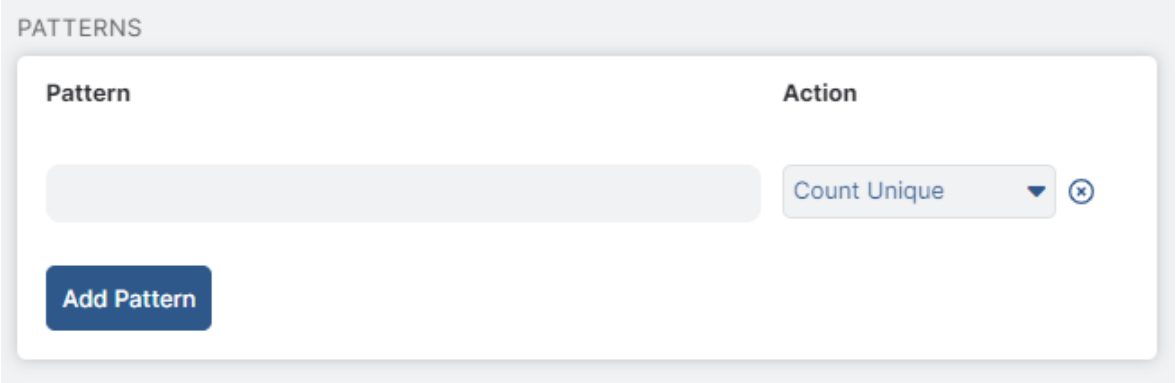
To learn more, see Adding Custom DLP Dictionaries.

Adding Patterns

To add patterns:

1. Go to **Administration** > **DLP Dictionaries & Engines**.
2. In the **DLP Dictionaries** tab, click **Add DLP Dictionary** or click the **Edit** icon for an existing dictionary.

   The **Add/Edit DLP Dictionary** window appears.
3. In the **Add/Edit DLP Dictionary** window:
   a. Enter the **Pattern** you want the dictionary to match. To learn more, see the guidelines and syntax requirements for patterns.
   b. Choose the **Action** the dictionary takes upon detecting valid matches:
      - **Count All**: The dictionary counts all matches of the pattern, including identical patterns, toward the match count. For example, you create a dictionary with a pattern for US phone numbers and choose Count All. When the dictionary scans content containing three instances of the same exact US phone number, it counts all three instances as three matches.
      - **Count Unique**: The dictionary counts each unique match of the pattern toward the match count only once, regardless of how many times it appears in the content. For example, you create a dictionary with a pattern for US phone numbers and choose Count Unique. When the dictionary scans content containing three instances of the same exact US phone number, it counts all three instances as one match.
4. To add another pattern, click **Add Pattern**.



5. Click **Save** and activate the change.

