CEL Funktionen

Theresa Henze - 2026-02-12 - Verschiedenes

Was ist CEL (Common Expression Language)?

CEL ist eine einfache, aber leistungsstarke Sprache zur Definition von Ausdrücken und Regeln. Dies wird häufig in Applikationen verwendet, um benutzerdefinierte Logik zu implementieren, ohne dass komplexer Code geschrieben werden muss. CEL ist besonders nützlich in Szenarien, in denen Flexibilität und Anpassungsfähigkeit erforderlich sind.

Bare.ID verwendet CEL in verschiedenen Bereichen:

- Nutzerprofil Validator (“CEL Programm”)
- Login Provider Mapper (“CEL Attribute Mapper”)
- Applikations-Mapper
    - für OIDC (“CEL claim”)
    - für SAML (“CEL claim”)

Übergreifende Hilfsfunktionen:

- `base64.encode(input: string) -> string`

  Gibt den Base64-kodierten Wert des Eingabewerts zurück.

- `base64.decode(input: string) -> string`

  Gibt den dekodierten Wert des Base64-kodierten Eingabewerts zurück.

- Die Makros aus CEL Sprachdefinition stehen ebenfalls zur Verfügung.

Die folgenden Abschnitte enthalten Beispiele für die Verwendung von CEL in den verschiedenen Bereichen.

Nutzerprofil Validator

Die Validierungsfunktionen verwenden CEL-Ausdrücke, um Eingabewerte zu überprüfen und Fehler zu melden.

Eingabewerte:

- `value: string`: Der Eingabewert, der validiert werden soll.
- `field: string`: Der Name des Feldes, das validiert wird.

Rückgabewert:

- `ValidatorResponse`: Ein Objekt, das eine Liste von Validierungsfehlern (`errors`) enthält. Ist die Liste leer, war die Validierung erfolgreich.

Hilfsfunktionen:

- `errorCase(condition: bool, message_key: string) -> ValidatorResponse`

  Gibt einen `ValidatorResponse` mit einem `ValidationError` zurück, wenn die Bedingung wahr ist, andernfalls einen leeren `ValidatorResponse`.

- `errorCases(conditions: map<string, bool>) -> ValidatorResponse`

  Gibt eine Liste von `ValidationError`-Objekten zurück, die den Bedingungen entsprechen, die wahr

sind.

Hinweis: Die Validierungsfunktionen werfen Fehler mit einem Übersetzungs-Schlüssel. Dieser Schlüssel muss in den Übersetzungen definiert werden, damit die Fehlermeldung korrekt für die verfügbaren Sprachen angezeigt wird. Siehe dazu das Handbuch Kapitel Texte.

Beispiele:

- Dies löst einen Fehler `input.not.test` aus, wenn der Eingabewert nicht `test` ist.

```
ValidatorResponse{
errors:
    [
        value == 'test' ? ValidationError{} : ValidationError{message_key:
'input.not.test'}
    ]
}
```

- Verkürzte Notation: Dies löst einen Fehler `error.expected.not.test` aus, wenn der Eingabewert `test` ist.

```
errorCase(value == 'test', 'error.expected.not.test')
```

- Mehrfache Fehler: Dies löst Fehler aus, wenn der Eingabewert
  - kleiner oder gleich 5 ist oder
  - größer oder gleich 10 ist.

```
errorCases({'error.lesser.than.expected': int(value) <= int(5),
'error.greater.than.expected': int(value) >= int(10)})
```

Login Provider Mapper

Prä-Prozessierung

Hier wird festgelegt, wie der Benutzername und die E-Mail-Adresse für den Login aus den vom externen Anbieter gelieferten Daten ausgewählt werden. So wird sichergestellt, dass die richtigen Informationen für die Anmeldung verwendet werden.

Dies geschieht vor Aufruf des “First Broker Login” Flows.

Eingabewerte:

- `brokered_identity_context: BrokeredIdentityContext`: Das Kontextobjekt, das die Informationen des externen Identitätsanbieters enthält.

```
BrokeredIdentityContext {
    string id
    string username
    string model_username
    string email
    string first_name
    string last_name
    string broker_session_id
    string broker_user_id
    String token
    google.protobuf.Struct context_data
}
```

Rückgabewert:

- `IdentityProviderPreprocessorResponse`: Ein Objekt, das Benutzername (`username`) und E-Mail-Adresse (`email`) des Benutzers enthält.

  ```
  IdentityProviderPreprocessorResponse{
      string username
      string email
  }
  ```

Beispiele:

- Mapping des Benutzernamens und der E-Mail-Adresse:

  ```
  IdentityProviderPreprocessorResponse{
      username: 'username',
      email: 'test@example.com'
  }
  ```

Mapping von Attributen und Zuweisung von Rollen und Gruppen

Eingabewerte:

- `brokered_identity_context: BrokeredIdentityContext`: Das Kontextobjekt, das die Informationen des externen Identitätsanbieters enthält.

  ```
  BrokeredIdentityContext {
      string id
      string username
      string model_username
      string email
      string first_name
      string last_name
      string broker_session_id
      string broker_user_id
      String token
      google.protobuf.Struct context_data
  }
  ```

Rückgabewert:

- `IdentityProviderMapperResponse`: Ein Objekt, das die Attribute und Gruppen- und Rollen-Zuordnungen enthält.

  ```
  IdentityProviderMapperResponse {
      AttributeEntry[] attributes
      string[] role_uuids
      string[] group_uuids
  }
  AttributeEntry {
      string key
      string[] values
  }
  ```

Hilfsfunktionen:

- `groups.listByName(search: string) -> GroupModel[]`

  Gibt eine Liste von Gruppen zurück, die dem Suchbegriff entsprechen.

  ```
  GroupModel {
      string id
      string name
      string full_path
  }
  ```

Beispiele:

- Mapping mehrerer Attribute und Zuweisung von Rollen und Gruppen:

```
IdentityProviderMapperResponse{
    attributes: [
        AttributeEntry{key: 'my-attribute', values: ['val1', 'val2']},
        AttributeEntry{key: 'broker-attribute', values:
[brokered_identity_context.id]}
    ],
    role_uuids: ['role-uuid-1'],
    group_uuids: ['group-uuid-1']
}
```

- Notation mit Hilfsfunktion:

```
IdentityProviderMapperResponse{
    group_uuids: groups.listByName('editor').map(g, g.id)
}
```

Applikations-Mapper

OIDC

Mapper für OpenID Connect (OIDC) Claims: Hier wird festgelegt, wie Benutzerattribute in den OIDC-Token eingebettet werden, die an Applikationen ausgegeben werden. Dies ermöglicht es, dass Applikationen die benötigten Informationen über den Benutzer erhalten.

Eingabewerte:

- `user_session: UserSessionModel`: Das Benutzer-Sitzungsmodell, das die Informationen des angemeldeten Benutzers enthält.

```
UserSessionModel {
    UserModel user
}
UserModel {
    string id
    string username

    GroupModel[] groups
}
GroupModel {
    string id
    string name
    string full_path
}
```

Rückgabewert:

- `OIDCProtocolMapperResponse`: Ein Objekt, das den Wert des Attributs (`claim_value`) enthält.

```
OIDCProtocolMapperResponse{
    string claim_value
}
```

Beispiele:

- Mapping eines booleschen Werts:

```
OIDCProtocolMapperResponse{claim_value: has(user_session.user.username)}
```

- Mapping eines numerischen Werts:

  ```
  OIDCProtocolMapperResponse{claim_value: size(user_session.user.username)}
  ```

- Mapping eines Text-Werts:

  ```
  OIDCProtocolMapperResponse{claim_value: user_session.user.id}
  ```

- Mapping einer Liste von Werten:

  ```
  OIDCProtocolMapperResponse{claim_value: [user_session.user.id,
  user_session.user.username]}
  ```

- Mapping einer Map bzw. eines JSON-Objekts:

  ```
  OIDCProtocolMapperResponse{claim_value: {'id': user_session.user.id, 'username':
  user_session.user.username}}
  ```

SAML

Mapper für SAML Attribute und NameID: Hier wird festgelegt, wie Benutzerattribute in die SAML-AttributeStatement eingebettet werden, die an Applikationen ausgegeben werden. Dies ermöglicht es, dass Applikationen die benötigten Informationen über den Benutzer erhalten.

Eingabewerte:

- `user_session: UserSessionModel`: Das Benutzer-Sitzungsmodell, das die Informationen des angemeldeten Benutzers enthält.

  ```
  UserSessionModel {
      UserModel user
  }
  UserModel {
      string id
      string username

      GroupModel[] groups
  }
  GroupModel {
      string id
      string name
      string full_path
  }
  ```

Rückgabewert:

- `SAMLProtocolMapperResponse`: Ein Objekt, das
  - **entweder** den Wert des Attributs (`attribute_value`) enthält
  - **oder** den Wert für die NameID (`mapper_name_id`) enthält.

    ```
    SAMLProtocolMapperResponse{
        oneof(
            string attribute_value
            string mapper_name_id
        )
    }
    ```

Beispiele:

- Mapping eines Attribut-Werts:

```
SAMLProtocolMapperResponse{attribute_value: 'test'}
```

- Mapping der NameID:

  ```
  SAMLProtocolMapperResponse{mapper_name_id: 'test'}
  ```