

Secure by Design: MidPoint Expression Language

Radovan Semančík
Software Architect
November 2026

Agenda

- Expressions
- The Past: Groovy
- The Present: MEL
- Tips & Tricks
- The Future: Secure by Default



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Expressions Provide Flexibility

- Magic dust that provides ultimate flexibility
- Mappings, object template,
  role autoassignment, policy rules,
  search filters, GUI customization, ...
- Scripting expressions:
  Heroes that saved a lot of projects
- Groovy as primary scripting language
  since midPoint 2.0 (2012)

```
givenName + ' ' + familyName

jobCode == 'X007'

'X' + input

basic.composeDnWithSuffix('uid', ...)

for ( a in focus.assignment ) {
  if (a.targetRef.type == ... ) {
     ...
  }
}
```

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Expressions Can Be Too Dangerous

- This magic dust provides a bit too much flexibility
- Groovy: built-in remote code execution (RCE) vulnerability
- “Industry standard” approach: ignore the problem
- Sandboxing does not really work
  - *default permit* vs *default deny* problem
  - upgrade problem
- Scripting languages are **not** designed for security

```
System.exit(0)

"${System.exit(0)}"

new File('/etc/passwd').text

'rm -rf /'.execute()

['rm','-rf','/'].execute()
```

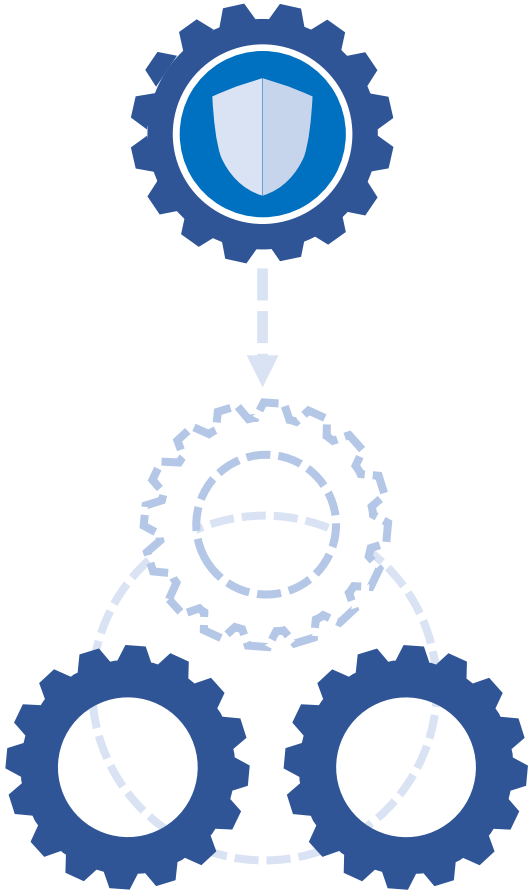
… and access to everything in Java runtime, all libraries, reflection … everything

© 2026 Evolveum s.r.o. All rights reserved.

Security by Design

- Security cannot be an afterthought,
  it has to be integral part of the design
- Mandated by CRA Article 13(2)
- Security is much more than “no vulnerabilities”

- We need language which is **designed to be secure**



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Looking for Secure Scripting Language

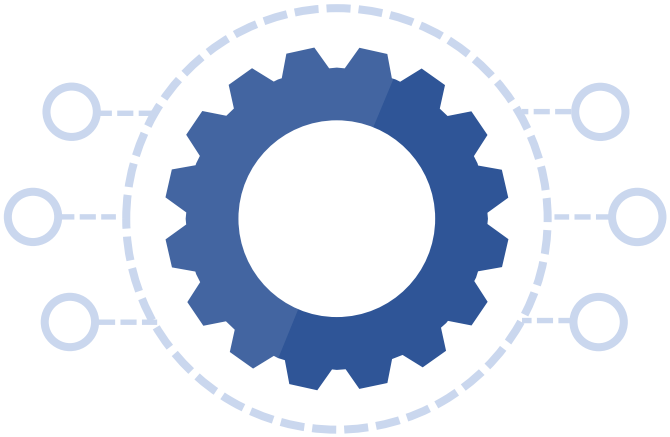
- Looking for language which is **designed to be secure**
- There is not much to choose from:
  - Rich languages → not secure (Groovy, Python, etc.)
  - Safe languages → not powerful (Liquid, Jakarta EL, etc.)
- Winner: **Common expression language (CEL)**
  by Google et al.
- However, CEL leaves much to be desired
- CEL is more what you'd call "framework" than actual language



© 2026 Evolveum s.r.o. All rights reserved.

MidPoint Expression Language (MEL)

- Based on Common expression language (CEL)
- Therefore, designed for security/safety
- Extended for use in midPoint
- MEL = CEL engine + midPoint extensions



```
focus.givenName + ' ' + focus.familyName

focus.activation.effectiveStatus == 'enabled'

matches(focus.telephoneNumber, '^[0-9]+$')
```

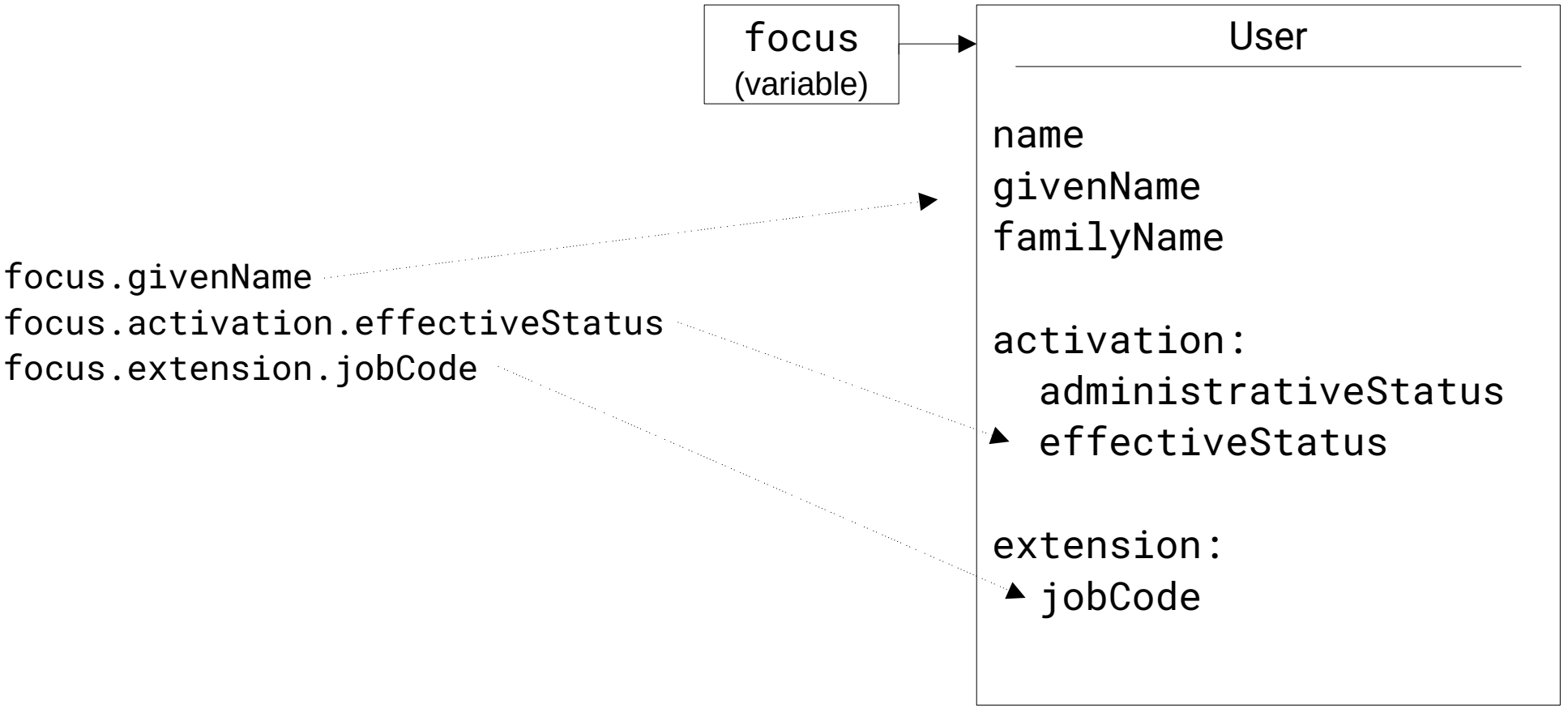
© 2026 Evolveum s.r.o. All rights reserved.

MEL Expressions in MidPoint

```
<expression>
    <script>
        <language>http://midpoint.evolveum.com/.../language#mel</language>
        <code>
            focus.givenName + ' ' + focus.familyName
        </code>
    </script>
</expression>
```

© 2026 Evolveum s.r.o. All rights reserved.

MEL and Structured Data



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL Operators and Functions

Operators

```
+ - * / % ! == != < <= > >= && || () ?: in . .? [] [?]
```

Functions (selection)

Global

```
size(x)
matches(s, regexp)
isEmpty(s)
contains(s, substring)
trim(s)
join(l, separator)
```

Member

```
x.size()
s.matches(regexp)
s.isEmpty()
s.contains(substring)
s.trim()
l.join(separator)
s.capitalize()
s.charAt()
```

© 2026 Evolveum s.r.o. All rights reserved.

MEL Specification

https://docs.evolveum.com/midpoint/reference/master/expressions/expressions/script/mel/specification/

MidPoint Expression Language Specification

4.11.0

Last modified 20 Oct 2026 08:42 UTC

MidPoint Expression Language (MEL) is an expression language designed to support customization of midPoint functionality in an easy-to-use and secure way.

This page provides detailed description of the language. It is meant to be a reference documentation of the language.

Literals

Literal	Prefix / Delimiter / Values	Example	Notes
String	' (apostrophe)	`'Hello world!'`	
Numeric	none	`42` `3.14`	
Boolean	`true` `false`	`true`	
Bytes	`b`	`b'\x68\x65\x6c\x6c\x6f'`	
Null	`null` `nil`	`null` `nil`	`nil` is dynamically-typed, see nullability notes.
List	[ ] (square brackets)	`[1, 2, 3]` `['foo', 'bar', 'baz']`	
Map	{} (curly brackets)	`{'one': 'unus', 'two': 'duo', 'thee': 'tres'}`	
Timestamp	`timestamp()` constructor function	`timestamp('2026-06-15T10:30:45.123Z')`	
Duration	`duration()` constructor function	`duration('1h30m')`	

Operators

Following *operators* can be used in MEL expressions:

Operator	Meaning	Notes

Table of Contents

- Literals
- Operators
- Variables
  - Local Variable Binding
- Functions and Macros
  - all
  - ascii
  - capitalize
  - charAt
  - contains(string)
  - contains(list)
  - containsIgnoreCase(string)
  - containsIgnoreCase(list)
  - debugDump
  - default
  - endsWith
  - equalsIgnoreCase
  - exists
  - exists_one
  - filter
  - format
  - has
  - indexOf
  - isBlank
  - isEmpty
  - isPresent
  - join
  - lastIndexOf
  - lc
  - list

©

MEL Examples

Username generator (simplified)

```
givenName.substring(0,1) + familyName
```

Role autoassignment expression

```
jobCode == 'X007'
```

Name formatting (e.g. custom column)

```
'%s, %s'.format([focus.familyName, focus.givenName])
```

Telephone number validator

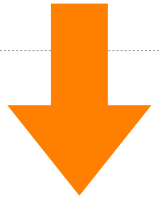
```
matches(focus.telephoneNumber, '^[0-9]+$')
```

© 2026 Evolveum s.r.o. All rights reserved.

MEL is an Expression Language

- MEL is an *expression* language
- Goal: return a value
- MEL is **not** a sequence of commands (not *imperative*)
- Evaluating values and functions (*functional* language)
- Functions are free of side effects (no “writes”)
- MEL has limitations, but it is still powerful

```
command;
command;
command;
command;
```



```
func(func(func(func())))
```

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Conditional

Groovy

```
if (input == null) {
  return 'foobar'
} else {
  return input
}
```



MEL

```
input == null ? 'foobar' : input
```



MEL (better)

```
default(input, 'foobar')
```

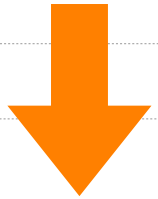
© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

No Loops

Groovy

```
for ( a in focus.assignment ) {
  if (a.targetRef.type == PolicyType.COMPLEX_TYPE) {
     return a.targetRef.oid
  }
}
```



MEL

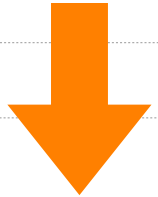
```
focus.assignment
  .filter(a, a.targetRef.type == 'PolicyType' )
  .map(a, a.targetRef.oid)
```

© 2026 Evolveum s.r.o. All rights reserved.

No Loops

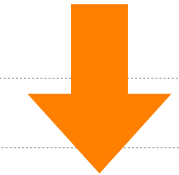
Groovy

```
for ( a in focus.assignment ) {
  if (a.targetRef.type == PolicyType.COMPLEX_TYPE) {
     return a.targetRef.oid
  }
}
```



MEL

```
focus.assignment
  .filter(a, a.targetRef.type == 'PolicyType' )
  .map(a, a.targetRef.oid)
```



MEL (shorter)

```
focus.assignment.map(a, a.isTargetPolicy(), a.targetRef.oid)
```

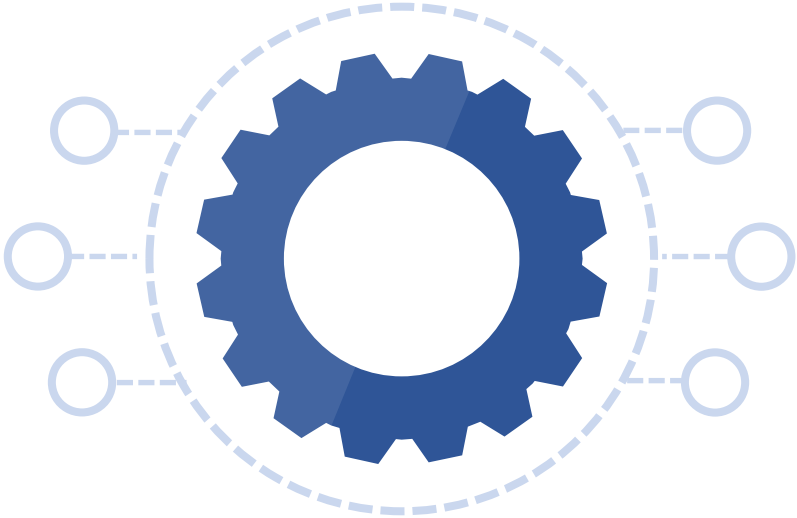
© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL Extensions

- Extra functions to make MEL more comfortable

- String/date formatting and parsing
- Logging and diagnostics
- Working with midPoint objects (e.g. shadows)
- Retrieving data from midPoint repository
  (read only – no side effects)
- LDAP data formats (e.g. DN)
- Secret providers



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL Extension: Format

```
format.concatName([focus.givenName, focus.familyName])

format.concatName(['John', 'Doe']) → 'John Doe'


format.strftime(timestamp, '%d/%m/%Y %H:%M:%S') → '04/11/2026 14:07:32'

format.strptime('04/11/2026 14:07:32', '%d/%m/%Y %H:%M:%S') → timestamp


format.parseGivenName('Ing. Radovan Semančík, PhD.') → 'Radovan'
```

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL Extensions

Log

```
log.debug('User {} was inactivated due to {}', [ focus.name, reason ])
```

MidPoint objects

```
projection.primaryIdentifiers() → ['rsemancik']
```

MidPoint functions

```
midpoint.getObject('UserType', oid) → user
```

LDAP

```
ldap.composeDnWithSuffix(['cn','foo','ou','baz','dc=example,dc=com'])
    → 'cn=foo,ou=baz,dc=example,dc=com'
```

© 2026 Evolveum s.r.o. All rights reserved.

MEL is Null-Safe

No explicit null checks are (usually) necessary

not necessary

```
focus.givenName == null ? null : focus.givenName.substring(0,1)
```



works reliably

```
focus.givenName.substring(0,1)
```

because

```
null.substring(0,1) → null
```

© 2026 Evolveum s.r.o. All rights reserved.

Strings and Polystrings

- Polystrings work just like strings
- Mix and match them as you like

```
focus.givenName.substring(0,1) + ' ' + focus.familyName
```

Polystring String String Polystring

```
focus.givenName == 'Radovan'
```

Polystring String

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Strings and QName

- We can pretend that QNames do not exist in MEL
- Strings can be used instead of Qnames and compared with QNames

```
focus.extension['jobCode']
```

expects QName

String

```
assignment.targetRef.type == 'RoleType'
```

QName

String

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL Recommendations and Remarks



- **Just write the code**
- Don’t worry about `null`, polystrings or qnames
- MEL and midPoint are going to figure it out for you
- However, some CEL-related quirks remain
- `stringify()` to the rescue:
  when in doubt, use `stringify()` to convert any value to non-null string
- Tips:
  https://docs.evolveum.com/midpoint/reference/master/expressions/expressions/script/mel/tips/
- Troubleshooting:
  https://docs.evolveum.com/midpoint/reference/master/expressions/expressions/script/mel/troubleshooting/

© 2026 Evolveum s.r.o. All rights reserved.

Real-World Examples

Username generator

```
focus.givenName.norm.replace(' ','').substring(0,1) +
focus.familyName.norm.replace(' ','').substring(0,7) +
iterationToken
```

Value presence check

```
isPresent(focus.activation.administrativeStatus)
```

Null-safe string formatting

```
'resource: ' + resource.name                        → resource: null

'resource: ' + default(resource.name, '(none)')     → resource: (none)
```

© 2026 Evolveum s.r.o. All rights reserved.

MEL Security

- MEL expressions do **not** have access to any Java runtime functionality, except those explicitly exposed through extensions.
- No access to files, network, processes, OS, midPoint internals, ...
- No direct way to cripple (DoS) the system,
  e.g. no infinite loops
- Secure enough for delegation,
  e.g. role auto-assignment, role engineering, reports, GUI customization
- Expression profiles are properly applied to allow/deny MEL extensions,
  e.g. `midpoint.*` and `secret.*` extensions denied by default
- MEL is secure in itself, yet expression result may still do damage (!!!)
  e.g. autoassign privileged role to all users



© 2026 Evolveum s.r.o. All rights reserved.

MEL Limitations

- Default midPoint configuration: MEL only
- MEL is not omnipotent – by design!
- Goal: All common “legal” use case should be possible in MEL
- When MEL is not enough:
  1. Contact us, we will extend MEL
  2. Resort to Groovy (temporary)

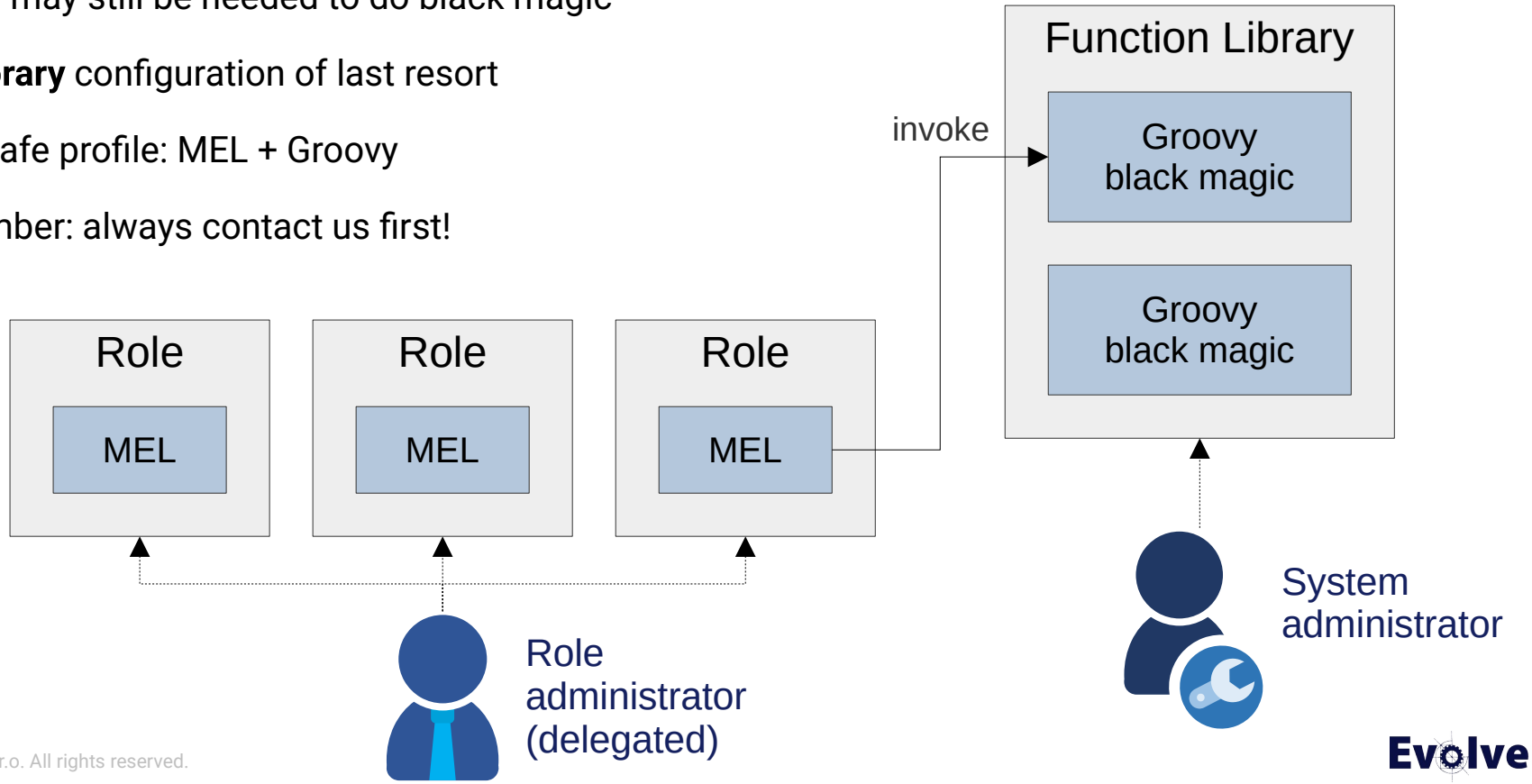


© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

MEL and Groovy

- Groovy may still be needed to do black magic
- **Temporary** configuration of last resort
- Semi-safe profile: MEL + Groovy
- Remember: always contact us first!



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Documentation, etc.

- MEL documentation:

  https://docs.evolveum.com/midpoint/reference/master/expressions/expressions/script/mel/

- MidPilot

  MidPilot can generate nice MEL mappings

- General-purpose AI chatbots

  They know CEL, they work fine with MEL

  Explicitly include documentation link in the prompt to let them know about MEL extensions (at least for now)

© 2026 Evolveum s.r.o. All rights reserved.

MidPoint 4.11 “Čapek”

- MidPoint 4.11 is the first release to support MEL
- MEL is the default scripting language in midPoint 4.11
- Groovy is disabled by default in midPoint 4.11
- Significant security improvement
  (hence the quick change)



Karel Čapek

© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Cybersecurity Made In Europe

- Founded by European Cyber Security Organisation (ECSO)
- European companies (ownership and R&D)
- Cybersecurity requirements (ENISA)

- Our commitment to EU legislation compliance (e.g. Cyber Resilience Act, AI Act, GDPR, ...)
- European digital sovereignty

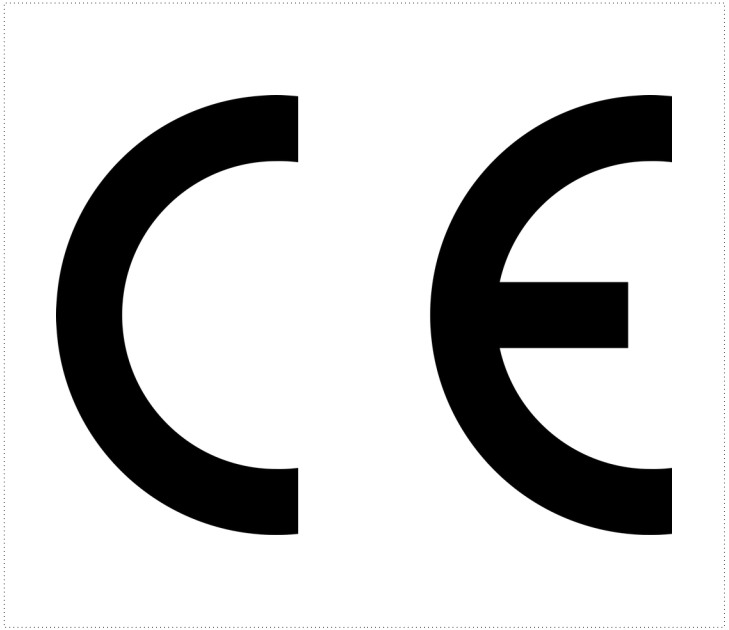


© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Cyber Resilience Act

- **Regulation (EU) 2024/2847** of the European Parliament and of the Council of 23 October 2024 on **horizontal cybersecurity requirements for products with digital elements** ...
- Products with digital elements: both hardware and software
- Essential cybersecurity requirements
- CE marking
- Full force in December 2027
- MidPoint 4.12 to be fully CRA-compliant



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

The Future: MidPoint 4.12 and Beyond

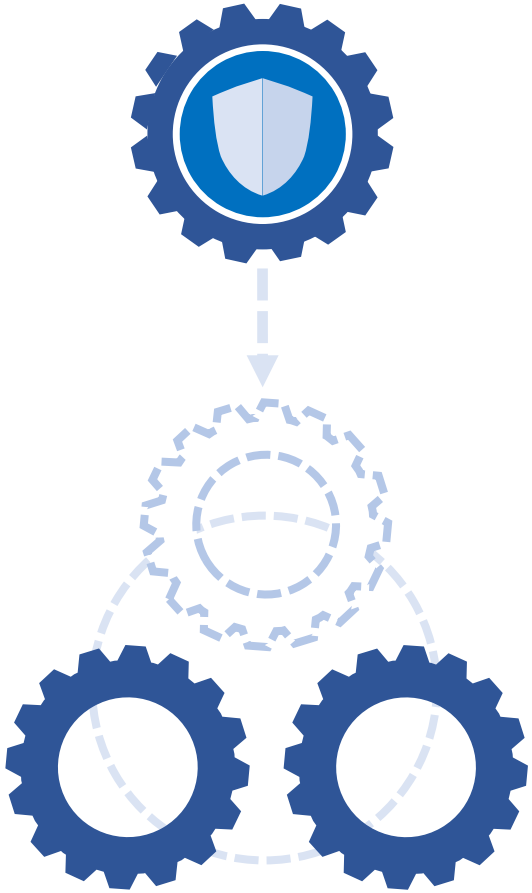
- Security by design
  - Native part of development processes
  - Transparent (public documentation)
- Secure by default
- Residual risk clearly communicated
- Commitment to full CRA compliance for midPoint 4.12
- CRA essential cybersecurity requirements satisfied
- Above and beyond in cybersecurity



Evolveum

Secure by Default

- Default configuration of midPoint must be secure
- Mechanism to easily maintain secure state,
  e.g. default roles, dashboards, policies, ...
- MEL is a major part of that effort
- Essential cybersecurity requirement
  CRA Annex I Part I (1)(b)
- Goal: midPoint 4.12 LTS



© 2026 Evolveum s.r.o. All rights reserved.

Evolveum

Conclusion

- Our commitment to improve and maintain cybersecurity
- MEL is a major step forward in midPoint 4.11
- Regulatory requirements (CRA)
- Secure by design, secure by default
- Instinctive cybersecurity → formal cybersecurity
- Planned for midPoint 4.12 LTS … and beyond
- Security effort never ends



© 2026 Evolveum s.r.o. All rights reserved.

Questions & Answers

Do you have any **questions**? Feel free to contact us at **info@evolveum.com**

Follow us on social media or **join us** at GitHub or Gitter!

/Evolveum /Evolveum /Evolveum /Evolveum

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

Evolveum

© 2026 Evolveum s.r.o. All rights reserved.