## Basic Syntax Rules

For a full list of syntax rules, see full MEL reference material.

| | Example | Description |
|---|---|---|
| #[ ] | #[message.id] | Always bounds an expression. |
| Simple expressions | [message.field]<br>[sessionVars.age | The simplest type of expression, these consist of just a context object and a field, or simply a variable. Provides access to information from the message including payload, properties, and variables. |
| Multi line expressions | #[calendar = Calendar.getInstance();<br>message.payload = ``neworg.mule.el.datetime.DateTime(calendar);] | You can include several lines in a single expression, each must end with a ; |
| Operators | #['Cookie' + flowVars.cookie] | Performs operations in expressions. Can be unary, comparison, logical, bitwise, arithmetic, and more. |
| Boolean expressions | #['foo'=='bar']<br>#[2 + 2 == 4] | Produces Boolean values. |
| Bean Property Access | #[payload.property1.property2] | Access information from bean. |
| Method invocations | #[message.header.get()] | Calls a method, then performs it on an object according to the parameter (if any) specified within the parentheses. Method calls always follow the syntax: object.method() |
| Assignments | #[payload = 'sample'] | Evaluates to assign a value. The example at left resolves dynamically to set the payload to sample. |
| Literals | 'expression'<br>255<br>null | Strings, numbers, Boolean values, types, and nulls. |

| | Example | Description |
|---|---|---|
| **xpath and regex** | xpath3('/orders/order[0]') | **xpath3** and **regex** provide ways of extracting context information that doesn't already exist as a single value. |
| **Wildcards** | wildcard("Hello*") | Matches a value (the message palyoad, by default) against a wildcard pattern, these use the metacharacters '?' to represent any single character and '*' for a repetition of any character. It's case sensitive by default. |

# Examples

There is really no such thing as a single *typical* MEL expression. That said, a few example expressions can help illustrate how MEL expressions resolve. As the following table of examples demonstrates, the values that MEL expressions return can be numerical values, logical values (`true` or `false`), strings, or virtually any data type. MEL expressions can also perform operations, invoke methods, and execute functions. Explore all the possibilities by consulting the complete syntax reference. Access full examples that illustrate how to use MEL expressions in applications.

| Expression | Description |
|---|---|
| #[2 + 2] | This expression evaluates to 4. |
| #[2 + 2 == 4] | This expression uses an operator to perform a comparison. It evaluates to true. |
| #[message] | This expression references a context object in MEL (`message`, `app`, `mule`, and `server`). The value of this expression is the message. |
| #[message.id] | This expression accesses the id field of the message context object. The value of this expression is the unique message id that Mule automatically assigns to the message. |
| #[payload.firstname] | This expression accesses an object within the field (payload) associated with the context object (message). If the object is a map item whose key is 'firstname' then this expression evaluates to the value associated with the key 'firstname'. If the object is a bean, the property will be returned. |
| #[payload[4]] | Same as above, but in this case – provided the field is a list – the expression returns the value of the 5th item in the list. (4 refers to the 5th item because the first item in the list is the 0 item.) |

| Expression | Description |
| --- | --- |
| #[message.header.get()] | This expression calls the "get" method and performs it on the object, according to the parameter (if any) specified within the parentheses. |

Dates:

- Return the system date and time in a dateTime object:

```
#[server.dateTime]
```

- Return current system time in nanoseconds as an integer:

```
#[server.nanoTime()]
```

- Return a dateTime object with the specified calendar and server locale:

```
#[calendar = Calendar.getInstance();
message.payload = new org.mule.el.datetime.DateTime(calendar);]
```

- Set the message payload to a Java calendar representation of the server date and time:

```
#[message.payload = server.dateTime.toCalendar()]
```

```
org.mule.DefaultMuleMessage
{
  id=ffd5d320-7e56-11e7-8554-701420524153
  payload=java.lang.String
  correlationId=<not set>
  correlationGroup=-1
  correlationSeq=-1
  encoding=UTF-8
  exceptionPayload=<not set>

Message properties:
  INVOCATION scoped properties:
  INBOUND scoped properties:
    accept=text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
    accept-encoding=gzip, deflate, br
    accept-language=en-US,en;q=0.8
    cache-control=max-age=0
    connection=keep-alive
```

```
    cookie=jenkins-timestamper-offset=-19800000;
csrftoken=C0ntag2hdDXC8eBvnqSPLpNYXIS98lWKbZLwytUoevgSsJXTMg5iS15yiacoFeUx; jenkins-
timestamper=system; jenkins-timestamper-local=false; hudson_auto_refresh=true
    host=localhost:8081
    http.listener.path=/abcd/
    http.method=GET
    http.query.params=ParameterMap{[]}
    http.query.string=
    http.relative.path=
    http.remote.address=/127.0.0.1:55968
    http.request.path=/abcd
    http.request.uri=/abcd
    http.scheme=http
    http.uri.params=ParameterMap{[]}
    http.version=HTTP/1.1
    upgrade-insecure-requests=1
    user-agent=Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/59.0.3071.115 Safari/537.36
  OUTBOUND scoped properties:
  SESSION scoped properties:
}
```