Model Initialization

An entity that is defined in the model can be implicitly initialized by using a statement for each entity type you want to initialize. The initialization is always in response to one or more events. If an incoming event is related to an entity that does not exist then an entity with the given identifier is created by the initialization.

An initialization statement cannot contain conditions, and actions are optional. For example, the following statement defines how an entity named `vehicle` is initialized from an event named `vehicle activity`.

```
a vehicle is initialized from a vehicle activity, where this vehicle comes from the
vehicle of this vehicle activity .
```

Defining an entity initialization in the business model from an event type causes the event to be routed to the runtime when it occurs. The routing happens regardless of whether the agents within the solution listen for this event type or not. If the bound entity exists the event is still routed.

The initialization of an entity happens before any rule agent reacts to an incoming event. However, when an event triggers the initialization of several entities, a rule might be executed before a remote entity is initialized. To make sure that all entities are initialized before a rule executes, you can add a condition in the rule so that it sends an event if the entity is null.

Java API entity initialization

When you implement an initialization by using the Java API, you use the Entity Initialization Extension wizard to create a dedicated `EntityInitializer` API class. The class is associated to an entity type by using the `@EntityInitializerDescriptor` annotation. The generated `EntityInitializer` class contains two methods.

`createEntityFromEvent(Event)`

The createEntityFromEvent(Event) method is called when a specific event type occurs, the business model contains an initialization statement for the entity type, and the bound entity does not exist. Use the method to create an entity instance or choose to create an instance of any subtype. You can overrule the request for the entity initialization in the business model by returning the value `null`. Or you can read the event and initialize the bound entity with data contained in the event.
The default implementation creates an uninitialized instance of the entity type that is defined in the business model.

Entity Enrichment

An entity that is defined in the business model can have attributes that are enriched from an external source.

A data provider to be used in the enrichment of an entity must be defined in the business model. The enrichment of an attribute must also be defined in the business model by using a rule-like statement. An enrichment statement includes the name of the data provider and the attributes to be returned. For example, the following statement defines the attributes `latitude` and `longitude` of the entity type `airport`, which are to be enriched by using the `weather provider` data provider.

```
an airport is enriched by the weather provider,
   given
      the latitude from the latitude of this airport,
      the longitude from the longitude of this airport,
   setting
      the temperature to the temperature of this weather provider,
      the wind speed to the wind speed of this weather provider,
      the precipitation to the precipitation of this weather provider.
```

The business model must include the following definition of the `airport` entity:

```
an airport is a business entity identified by an IATA code with
   a latitude (a number),   a longitude (a number),
   a precipitation level (a number),
   a wind speed (a number),
   a temperature (a number).
```

The data provider is defined by specifying a name, the required attributes, and the returned data.

```
a weather provider is a data provider,
   accepts
      a latitude (a number),
      a longitude (a number),
   returns
      a temperature (a number),
      a wind speed  (a number),
      a precipitation  (a number).
```

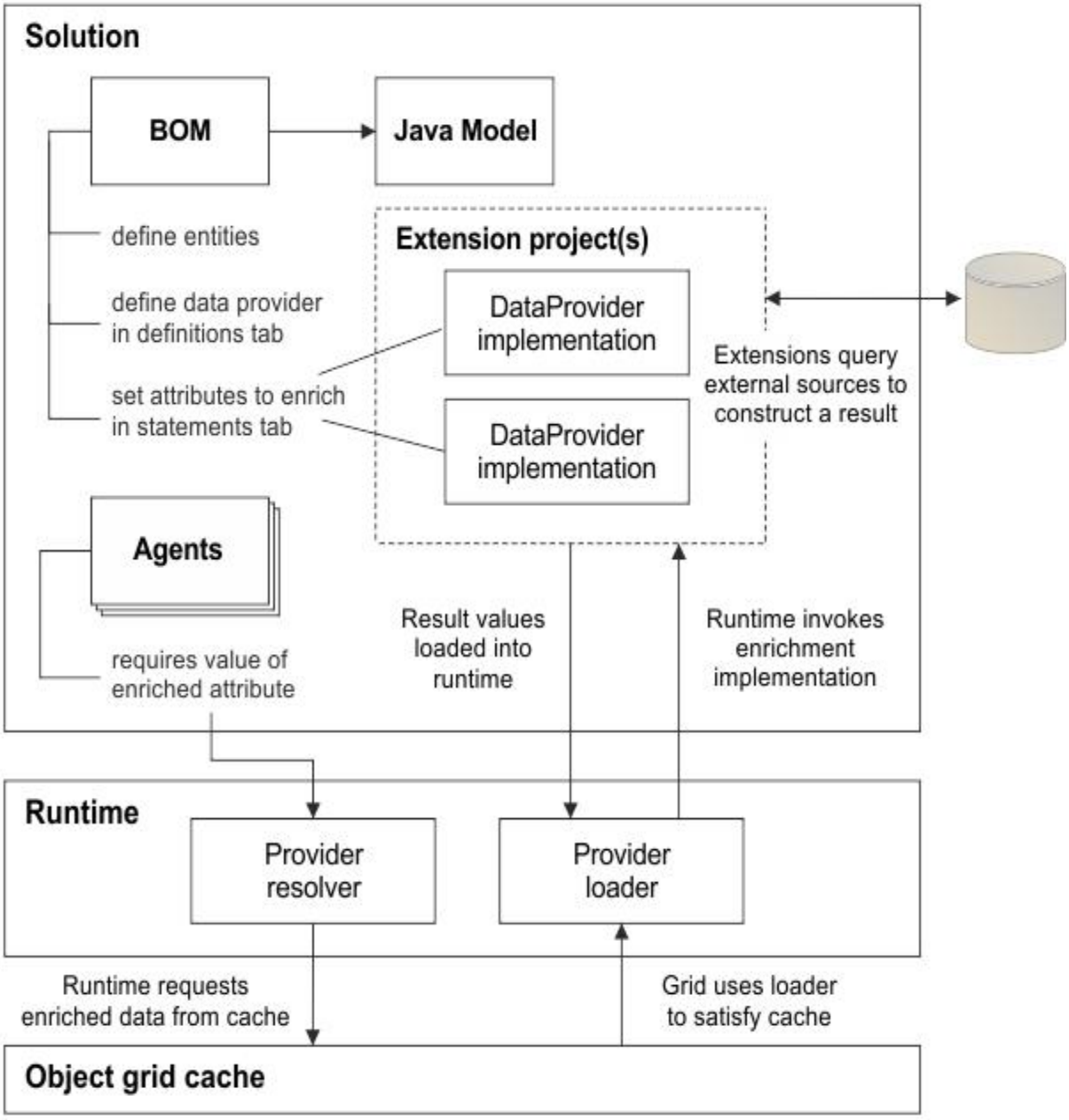
When an agent requests the value of `temperature`, `wind speed`, or `precipitation` of the `airport` entity the runtime calls the associated `weather provider`. The values of the returned attributes are cached by default during a transaction. The values can be cached for longer than a single transaction to reduce the cost of repeated calls to the data provider.

Note: A single entity can be enriched by multiple providers. One or more input values can be specified , which can be any attribute of the entity, including attributes of attributes. You cannot enrich a single attribute with more than one data provider.

When you implement the data provider, you use the Data Provider Extension wizard to create a dedicated DataProvider API class. The `DataProvider` class provides the getConceptFactory method to return complex attribute values. The default implementation uses the `@DataProviderDescriptor` annotation to bind the implementation to the provider name that is defined in the business model statements.

The `DataProvider` implementation is deployed into an extension project that is part of the solution. A complete solution must contain one implementation for each provider that is named in the business model enrichment statements. And each data provider name must be unique within the solution.

The following figure shows how the enrichment of data is defined in the business model. The runtime resolves and loads a data provider implementation when an agent requires an enriched attribute value (by referring to the attribute in a rule, for example). The grid caches and shares the data between transactions.


Solution
BOM
Java Model
define entities
define data provider in definitions tab
set attributes to enrich in statements tab
Extension project(s)
DataProvider implementation
DataProvider implementation
Extensions query external sources to construct a result
Agents
requires value of enriched attribute
Result values loaded into runtime
Runtime invokes enrichment implementation
Runtime
Provider resolver
Provider loader
Runtime requests enriched data from cache
Grid uses loader to satisfy cache
Object grid cache